(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 317 109 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
04.06.2003 Bulletin 2003/23

(51) Int Cl.7: **H04L 29/06**, G06F 17/30

(21) Application number: **01128541.8**

(22) Date of filing: **29.11.2001**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(71) Applicant: **Sony International (Europe) GmbH
10785 Berlin (DE)**

(72) Inventors:
- **Röhrle, Klaus,
Sony International (Europe) GmbH
70327 Stuttgart (DE)**
- **Schramm, Oliver,
Sony International (Europe) GmbH
70327 Stuttgart (DE)**
- **Bosau, Detlev, University of Stuttgart
70565 Stuttgart (DE)**

(74) Representative: **Rupp, Christian, Dipl.-Phys. et al
Mitscherlich & Partner,
Patent- und Rechtsanwälte,
Postfach 33 06 09
80066 München (DE)**

(54) System and method for controlling the adaptation of adaptive distributed multimedia applications

(57) The underlying invention generally relates to the field of Distributed Multimedia Systems (DMSs), applications and technologies in heterogeneous communication networks with highspeed access. It includes research and development issues which are especially related to multimedia middleware, Quality-of-Service (QoS) Management, adaptive applications, mobile terminals and wireless communication. In this context, the invention presents a sophisticated middleware framework (304) implemented on a client (204) supporting distributed multimedia applications (302) to dynamically adapt to varying resource availability. The middleware framework (304) enables the distributed multimedia applications (302) to specify the media they want to use and the relationships between these media. Using this information from all running distributed multimedia applications (302), the middleware framework (304) calculates the adaptation possibilities and controls the adaptation process.

Said middleware framework (304) comprises a Media Management Subsystem (404) for managing multimedia data to periodically send status events from running media items (614), which contain measured information of media and transmission parameters, such as frame rate and/or packet loss, and an Adaptation Engine (402) for controlling the adaptation of said distributed multimedia applications (302) to the current resource availability.

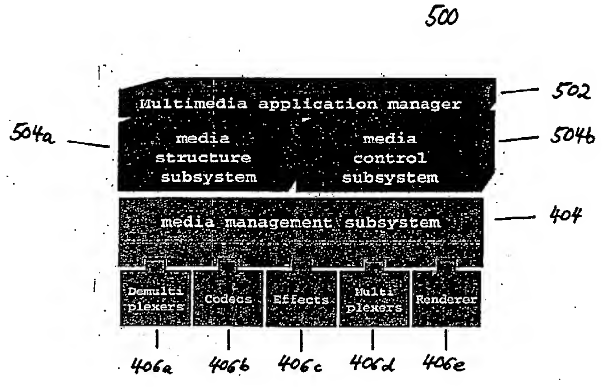

FIG. 5

Printed by Jouve, 75001 PARIS (FR)

Description

FIELD AND BACKGROUND OF THE INVENTION

[0001] The underlying invention generally relates to the field of Distributed Multimedia Systems (DMSs), applications and technologies in heterogeneous communication networks with high-speed access. It includes research and development issues which are especially related to multimedia middleware, Quality-of-Service (QoS) Management, adaptive applications, mobile terminals and wireless communication. In this context, the invention presents a sophisticated middleware framework implemented on a client supporting distributed multimedia applications to dynamically adapt to varying resource availability. The middleware framework enables the applications to specify the media they want to use and the relationships between these media. Using this information from all running applications, the middleware framework calculates the adaptation possibilities and controls the adaptation process.

[0002] The integration of multimedia into a distributed computing environment opens up a large spectrum of possibilities for new and exciting applications. Since multimedia is relevant to a wide range of application domains, such as entertainment, information retrieval and business systems, and is potentially deployable on a magnitude of end systems, the problem of supporting multimedia is a substantial one. In a distributed environment, the structural description of a multimedia document can be stored apart from the media object content. Thereby, the applications reside on (mobile) clients but access or update information from remote servers using (wireless) networks.

[0003] Generally, multimedia data can be categorized into one of two classes: discrete or continuous. Discrete data, usually in the form of text and images, represents a sequence of individual elements which are time-independent. The resources required to support this form of data are often minimal, and can easily be computed a priori. On the other hand, continuous data (which includes audio, video and animation) represents information which incorporates an inherent temporal dimension that is directly relevant to the user's perception of the information.

[0004] Continuous media are particularly demanding on processing and communication resources, generally due to the large amount of information involved and its temporal sensitivity. Hence, a large part of continuous media processing is carried out by means of specialized peripherals and end system hardware technologies. Thereby, high-performance networking technologies, such as the Asynchronous Transfer Mode (ATM), provide sufficiently high bandwidths for the transmission of continuous media streams. In general, continuous media streams such as video and audio can be transmitted and presented in different qualities, and the quality desired by the user or offered by an information provider largely influences all system parts, such as end systems and routers.

[0005] In this context, the design of distributed multimedia applications, such as systems providing an access to remote multimedia databases or teleconferencing, requires a careful consideration of QoS monitoring and resource management issues to provide end-to-end QoS guarantees for a large number of applications. Increased platform heterogeneity and varying resource availability in distributed systems motivate the design of resource-aware applications, which ensure a desired performance level by continuously adapting their behavior to changing resource characteristics. Thereby, the presentation quality, varying network load and stringent real-time requirements of continuous media streams require a relatively high utilization of networking bandwidth and processing power in the end systems.

[0006] For applications running in a shared environment which rely on the transfer of continuous media data, the allocation and management of these resources is an important question. It is essential that QoS is assured system-wide, including the application, the middleware, the operating system and the network. However, the majority of conventional systems is based on best-effort approaches. In general, these best-effort approaches are not suitable for DMSs because some users may be ready to pay higher prices for obtaining a maximum quality, while others may prefer low-cost presentations with lower quality. Additionally, for a teleconferencing application involving many users, a single QoS level may not be appropriate for all participating users, since some users may participate with a very limited local workstation which cannot provide the quality which is adopted by the majority of the conference participants. Therefore, distributed multimedia applications must be provided with different levels of quality, often corresponding to different levels of cost.

[0007] To date, much work on QoS management has been done in the context of high-speed networks providing Asynchronous Transfer Mode (ATM) in order to guarantee a specific QoS for the provided communication service characterized by the bandwidth of the media stream and the delay, delay jitter and packet loss rate provided by the network. Further QoS application requirements include:

- performance requirements (e.g. timeliness of execution and relative processing speed requirements),
- reliability requirements (e.g. high availability and relative failure recovery requirements), and
- security requirements (e.g. authentication and privacy).

[0008] In general, two possible ways exist to cope with resource management in complex heterogeneous DMSs:

- First, guaranteed resource allocation can be maintained and enforced in spite of the cost in terms of complexity and overhead.
- Second, applications (and other system components) can be developed which are more tolerant to inevitable fluctuations in the supporting environment's performance. This leads to the notion of "adaptive applications" designed to cope with unexpected QoS variations due to dynamically changing resource situations.

[0009] Despite the apparent novelty of the term "adaptive application", conventional configurable and tailorable programs according to the state of the art let users activate only modules which are needed in a particular context. Thereby, programs can be adapted to their hardware and operating system environment, e.g. to the available memory size, the presence of a floating-point coprocessor, or the I/O environment. These programs often automatically recognize the available resources (such as displayed size and type); if not, an adaptation is done manually at installation time. Other types of adaptivity include option negotiation for recognizing the parameters of remote components (such as terminals and modems), and program reconfigurability in fault-tolerant systems. Furthermore, error detection and correction in conventional communication protocols as mechanisms to adapt these protocols to changing transmission QoS can be considered.

BRIEF DESCRIPTION OF THE PRESENT STATE OF THE ART

[0010] According to the state of the art, different methods and technologies are currently available concerning QoS Management in DMSs, which are closely related to the topic of the underlying invention. In order to understand the context of the invention, it is necessary to briefly explain the main features involved with said technologies.

[0011] Today, the processing of multiple streams is generally facilitated by means of a resource sharing, managed by the operating system. This results in the potential to offer a wide range of QoS properties that define both how the end-user perceives the multimedia information and how the operating system should manage the associated end system and network resources. This characteristic is important since there is a finite limit to the number of multimedia streams the end system and network resources can successfully handle. Therefore, when end system or network resources are stretched to near capacity, policies, defined by either the end-user or an administrator, are used to automate share balancing.

[0012] Many multimedia data-types, particularly video, are processed and rendered through specialized end system devices. In parallel with the progression of multimedia, there have also been extensive advances in network communications technology. Fast-Ethernet (100 MBit/s) and Gigabit-Ethernet now enable relatively inexpensive communication in the field of Local Area Networks (LANs). Asynchronous Transfer Mode (ATM) technologies offer a variety of network bandwidths from 25 MBit/s to the desktop and 2.4 GBit/s in the backbone. Furthermore, ATM can provide firm QoS guarantees on network traffic catering for media types that are resource-sensitive, such as audio and video. Similar advances can be seen in the area of public wireless networks with upcoming technologies like GPRS (115 Kbit/s) and UMTS (2 Mbit/s) and wireless local area networks like WaveLan (54 Mbit/s) or HyperLAN (54 Mbit/s). The later one also includes QoS Management. Combined together, the advances in multimedia and network communications enable the deployment of sophisticated distributed multimedia applications. The deployment of interactive television, electronic publishing, tele-shopping, tele-medicine, tele-education, entertainment and gaming applications is now feasible. In reality, however, their success is impaired by an insufficient standardization across different technologies, coupled with a general lack of resources and support for distributed application development.

[0013] To address the problems of insufficient QoS management support, the concept of "middleware", a distributed homogeneous layer composed of generic support in the form of software libraries and services, was introduced. Originally, middleware was used to represent an enabling technology for client-server applications, allowing a client application to access heterogeneous database servers. It now represents a wide range of value-added software services that lie above the operating system and beneath the application. For this reason, services at this level are able to directly interact with each other. Thereby, in order to assure higher-level QoS, it is often necessary to utilize resource management mechanisms deployed within the operating system layers. However, conventional middleware technologies like CORBA, DCOM and Java RMI cannot provide QoS guarantees, although they have proved adequate for the development of elastic (as opposed to real-time) distributed applications.

[0014] Middleware services supporting QoS-aware applications can be provided according to one of the following two principal approaches; namely, the reservation-based and the adaptationbased approaches. The first approach requires to implement complex resource reservation, admission control and resource negotiation mechanisms. In contrast, the second approach requires to implement mechanisms that periodically monitor the resource availability of the execution environment, and provide applications with appropriate adaptation and reconfiguration properties.

[0015] Further activities in the field of adapting distributed multimedia applications to resource availability can be taken from the academic work of the authors Fangzhe Chang and Vijay Karamcheti. In their dissertation "A Framework for Automatic Adaptation of Tunable Distributed Applications" (published in Cluster Computing: The Journal of Networks, Software Tools and Applications, 2001) the authors focus on automatically adapting distributed applications to changes in the characteristics and availability of resources in dynamic, heterogeneous execution environments. The adaptation trades off among different resource requirements, application configurations, and output qualities based on user preferences to produce a desired execution behavior. Thereby, a framework is proposed that exposes alternate forms of application composition and execution, acquires the resource usage profiles of the application on a virtual execution environment, monitors the underlying resource availability, and automatically configures the application in response to changes of resource conditions. The disclosed framework has been applied to both distributed and parallel applications and shows significant improvement in performance and satisfaction of user preferences.

[0016] A parallel work has been conducted in the "Broadband Radio Access for IP-based Networks" (BRAIN) project. Within that context, the main aspect was not to support adaptation control but to create a QoS framework offering resource guarantees for multimedia streams.

PROBLEMS TO BE SOLVED BY THE INVENTION

[0017] Multimedia applications request timely transmission of their data. This translates into the requirement that networks provide a sufficient Quality of Service (QoS). This is especially difficult in wireless networks as the QoS in these networks (e.g. bandwidth, delay, jitter) varies on a large scale.

[0018] To cope with this situation, multimedia applications should be adaptive. They should be able to operate based on best-effort environments and attempt to adapt to a varying resource availability for the purpose of providing the best possible experience for the user on the available resource conditions, and achieving the most graceful quality degradation in case of scarce resources. Fig. 1 exhibits an example 100 of an adaptive application for adapting two multimedia scenarios 102 and 104 with different bandwidths and different QoS requirements.

[0019] In the scenario 200 depicted in Fig. 2, the adaptation control is done by the client 204, as the client 204 knows best about its resources and how it wants to utilize them. The involved servers 206 and 208 are able to support the adaptive decisions of the client 204 by providing data at various qualities. Resources taken into account for the adaptation are local resources like CPU and memory as well as network resources such as the available bandwidth.

[0020] Thereby, as can be taken from the client architecture 300 depicted in Fig. 3, it is advantageous to introduce a middleware framework 304 on the client 204 to support adaptive multimedia applications 302 due to several reasons:

- The middleware framework 304 can coordinate different applications 302 running on the client 204.
- Without any support, each application 302 would have to program such functionality on its own - which is not very efficient.
- The middleware framework 304 does not require tight integration or modifications in the kernel of the employed operating system and in the network protocol stack.

[0021] Some major problems such a middleware framework 304 has to solve are:

- to allow applications 302 to describe their adaptation possibilities, and
- to control the adaptation process (for all running applications) in such a way that, on any resource conditions, the adaptation process is steered towards maximum achievable user satisfaction.

OBJECT OF THE UNDERLYING INVENTION

[0022] In view of the explanations mentioned above, it is an object of the underlying invention to propose a novel technology supporting Distributed Multimedia Systems (DMSs), applications and technologies in heterogeneous communication networks with high-speed access to dynamically adapt to varying resource availability.

[0023] This object is achieved by means of the features of the independent claims. Advantageous features are defined in the dependent claims. Further objects and advantages of the invention are apparent in the following detailed description.

SUMMARY OF THE INVENTION

[0024] The proposed solution is basically dedicated to a sophisticated middleware framework implemented on a client targeted to support distributed multimedia applications to dynamically adapt to varying resource availability. Thereby, said framework calculates the adaptation possibilities and controls the adaptation process using information from all running applications. It enables the applications to specify the media they want to use and the relationships between these media. Furthermore, the framework is specialized enough to offer a lot of support functionality for the employed applications, but still general enough to support a wide range of applications. Parts of the framework can also be used in other application domains, in which the behavior of an application has to be controlled based on a set of measured parameters.

[0025] An architecture 400 of a middleware framework 304 as depicted in Fig. 4 which is able to perform the described tasks needs at least two parts:

- a media handling system for managing multimedia data, e.g. the Java Media Framework, DirectShow, Quicktime, or MPEG-21, and

- a component for controlling the adaptation, a so-called "Adaptation Engine", having various links to the media handling subsystem, e.g. to obtain measurements of media parameters.

[0026] In the following, the focus will be on the second part, assuming that a media handling system is in place.

BRIEF DESCRIPTION OF THE CLAIMS

[0027] The independent claim 1 and the dependent claims 2 to 23 refer to a middleware framework implemented on a client targeted to support distributed multimedia applications in heterogeneous communication networks to dynamically adapt to varying resource availability. Thereby, said middleware framework allows each running distributed multimedia application to specify the media it wants to use and the relationships between these media, calculates the adaptation possibilities, and controls the adaptation process.

[0028] Next, the dependent claim 24 is directed to a method for operating a middleware framework according to anyone of the claims 1 to 23.

[0029] Moreover, the dependent claims 25 and 26 pertain to a client connected to a client-server architecture of a distributed computing environment, which comprises a middleware framework according to anyone of the claims 1 to 23, and performs a method for operating a middleware framework according to claim 24.

[0030] In the dependent claims 27 to 32, a client system consisting of different client terminals communicating over a network is disclosed, in which one part of the resources is controlled by at least one Central Resource Controller for all client terminals.

[0031] Finally, the dependent claims 33 and 34 refer to a multimedia document browser for running multimedia retrieval applications on a client terminal, in which a middleware framework according to anyone of the claims 1 to 23 and a method for operating a middleware framework according to claim 24 is applied.

BRIEF DESCRIPTION OF THE DRAWINGS

[0032] Further advantages and possible applications of the underlying invention result from the subordinate claims as well as from the following description of the preferred embodiment of the invention which is depicted in the following drawings:

Fig. 1	shows an example 100 of an adaptive application for adapting two multimedia scenarios 102 and 104 with different bandwidths and different Quality of Service (QoS),
Fig. 2	illustrates an example 200 for a multimedia retrieval application with distributed servers 206 and 208, and one client 204 connected to the Internet 202,
Fig. 3	shows a client architecture 300 comprising a middleware framework 304,
Fig. 4	presents an overall architecture 400 of the middleware framework 304 comprising an Adaptation Engine 402 and a Media Management Subsystem 404,
Fig. 5	shows a detailed block diagram 500 of the overall architecture 400 of the middleware framework 304,

Fig. 6 shows an example 600 of a media structure tree,

Fig. 7 exhibits an overview of an architecture 700 comprising a Media Structure Subsystem 504a (MSS),

Fig. 8 exhibits a block diagram 800 showing the interfaces 802b of the Media Control Subsystem 504b (MCS) to other subsystems 404, 504a, 804 and 806,

Fig. 9 presents an overview 900 of the Media Control Subsystem 504b (MCS),

Fig. 10 shows an example 1000 of a client consisting of different terminals 204 and a central communication management 1002, called Central Resource Controller (CRC),

Fig. 11 shows an example 1100 of distributed client terminals 204 with two Central Resource Controllers 1002 (CRCs), and

Fig. 12 presents an example 1200 of an application with a multimedia document browser 1202 using the middleware framework 304.

DETAILED DESCRIPTION OF THE UNDERLYING INVENTION

[0033] In the following, the preferred embodiment of the underlying invention as depicted in Figs. 5 to 12 shall be explained in detail.
[0034] The Adaptation Engine 402 as depicted in Fig. 4 consists of two main subsystems 504a+b, as can be taken from the block diagram 500 in Fig. 5:

- the Media Structure Subsystem 504a, which gets information from the applications 302, and aggregates this information to a list of possible configurations - the so-called "adaptation space", and
- the Media Control Subsystem 504b implementing a control loop which dynamically selects a configuration based upon the current resource availability, implements it, and notifies the affected applications 302.

[0035] In the following, the term "media item" shall be used as an abstraction for different kinds of media. Furthermore, it shall be distinguished between "discrete media items" (e.g. images) and "continuous media items" representing time-based media (e.g. audio or movie clips). Sometimes, continuous media items can be controlled concerning their quality and resource consumption. These media items 614 shall be called "controllable media items". Controllable media items 614 offer a set of configurations, each resulting in a different resource consumption. An user of controllable media items 614 can dynamically switch between different configurations. Controllable media items 614 offer information about the configurations they support, including the resource consumption of each configuration. The resource consumption for a resource can be specified using an exact value, or by specifying a value range.
[0036] In the following, the main components of the Adaptation Engine 402, as depicted in Fig. 5, shall be described in detail.

1. Multimedia Application Manager

[0037] A Multimedia Application Manager 502 (MAM) is a component which is responsible for offering the Application Programming Interface (API) of the Adaptation Engine 402 to the applications 302. When using the Adaptation Engine 402, the applications 302 will primarily interact with the MAM 502. Therefore, it acts as a wrapper 802a for the other components of the Adaptation Engine 402.
[0038] The multimedia application 302 also manages the other subsystems of the Adaptation Engine 402. During start-up it builds and connects the contained subsystems. During shut-down it calls the corresponding methods to decouple and to deallocate the other subsystems.
[0039] If different implementations of a subsystem exist, the MAM 502 can be configured which one it should use.

2. Media Structure Subsystem (MSS)

[0040] To allow the middleware framework 304 to automatically make adaptation decisions, applications 302 have to transfer knowledge about the involved media items 614 and the requirements and restrictions for the usage of these media items 614 to the middleware framework 304.
[0041] Using the MSS API, a multimedia application 302 can specify the media items 614 it wants to use and the

relationships among these media items 614 (together called a "media structure"). Relationships between media items 614 are specified by rules like "Play media item A, B and C in parallel."
[0042] Thereby, the following rule classes can be distinguished:

- Aggregates that combine multiple media items 614 into so-called "media elements": Thereby, a media element can itself be a child of another media element. The resulting structure is a tree with the media items 614 as leaves, as can be taken from Fig. 6.

- Media element attributes 618 which are associated to a media element: One important attribute is the "priority" of a media element for an application 302. An important type of attributes 618 are constraints for the media element. For example, they make it possible that the contained media items 614 do not exceed a certain bandwidth.

- Media item attributes 618 which are associated to a media item: In contrast to media element attributes 618, media item attributes 618 additionally includes media parameters, e.g. for a movie clip the codec, the duration, the size and others. Thereby, controllable media items 614 describe their controllable parameters using media item attributes 618.

- Additional constraints being similar to the ones given as attributes 618 but not associated to a media item 614 or media element: These constraints can be created between any number of media items 614 and media elements.

[0043] The applications 302 communicate their rules by creating corresponding data structures that are added to a global data structure managed by the middleware framework 304. To create such a data structure, the applications 302 need to create media items 614, aggregates, add attributes 618 to these aggregates, combine the aggregates to an aggregate tree 600 as depicted in Fig. 6, add additional constraints, and commit the resulting data structure to the middleware framework 304.
[0044] Thereby, the applications 302 can use the following aggregates:

- The <sync> node 610 specifies that the contained media items 614 should be played in a synchronized manner.

- The <par> node 608 specifies that all contained media elements should be played in parallel.

- The <seq> node 609 specifies that all contained media elements should be played sequentially.

- The <switch> node 612 specifies that the contained media elements can be played alternatively. It is the most important aggregate for an adaptation as it allows the application 302 to define different adaptation possibilities. Moreover, a <switch> node 612 may contain an empty alternative.

[0045] The MSS 504a itself additionally manages two other aggregates:

- The <app> node 606 contains all media structures of one application 302.

- The <user> node 604 contains all media structures of one user.

- The <root> node 602 is the root of the media structure which contains all other aggregates known to the system.

[0046] Other aggregates, e.g. representing other groups, may be added. If one Adaptation Engine 402 is used to make an adaptation decision for a set of client systems, these systems may also be modeled by corresponding aggregates.
[0047] As a result, the MSS 504a manages one media structure tree 600 for the whole system. The tree 600 consists of media elements and media items 614 both with associated attributes 618. On top of this structure, additional constraints can be added which have connections to at least two nodes of the tree 600 (media elements or media items 614). The tree 600 changes dynamically as the applications 302 add or remove media structure information.
[0048] The following grammar in Extended Backus-Naur Form (EBNF) notation shows the rules on how to build a media structure tree 600:

```
<root> ::= <user> {<user>} <attributes>
<user> ::= <app> {<app>} <attributes>
<app> ::= <media group> {<media group>} <attributes>
```

```
<media group> ::= <media element>
      <media item> <attributes>
<media element> ::= <par> I <switch> I <seq> I <sync>
<par> ::= <media element> I <media item>
      {<media element> I <media item>}
<switch> ::= <media element> I <media item>
      {<media element> I <media item>}
<seq> ::= <media element> I <media item>
      {<media element> I <media item>}
<sync> ::= <media item> {<media item>}
<media item> ::= <media locator> <media attributes>
```

[0049] Not shown in this grammar are the definitions of media locator, media attributes and attributes 618. Media locators can be Uniform Resource Locators (URLs), but also other addressing mechanisms can be used. Here, the attributes 618 are not specified as for each aggregate arbitrary attributes 618 can be used (key-value pairs). The grammar also does not show the possibility to add constraints. Constraints can be added to any existing set of aggregates.

[0050] The first three rules are used only by the middleware framework 304 itself. <user> nodes 604 are created and removed whenever the user logs in or out of the system. <app> nodes 606 are created whenever an application 302 starts, and removed when the application 302 stops. The middleware framework 304 can also add constraints either automatically to reflect system or regulatory constraints, or on user requests.

[0051] The other rules are used by the applications 302. Each time an application 302 commits a media structure to the middleware framework 304, it is added as a child to the corresponding <app> node 606. Thereby, the applications 302 can dynamically add media structures.

[0052] The resulting data structure is a graph containing information from the applications 302 and from the middleware framework 304. Each node in the graph can have additional information in the form of attributes 618.

[0053] One of the main tasks of the MSS 504a is to calculate a list of alternative configurations by using the media structure. This list is needed by the Media Control Subsystem 504b (MCS) to make adaptation decisions. It therefore must contain all information needed to make those decisions.

[0054] A configuration is defined by a set of media items 614 which have to be played in parallel plus a list of attributes 618. A list of configurations therefore describes different choices that result from the evaluation of the media structure. Therefore, the configuration list is a description of the adaptation space.

[0055] For calculating the configuration list, the MSS 504a takes the information from the <switch> aggregates 612 and the different configurations of controllable media items 614 into account. This means that two configurations differ either in at least one media item 614 or in one configuration of a controllable media item 614.

[0056] The MSS 504a further associates a list of attributes 618 to each configuration (such as a priority, which is calculated using the priorities from the aggregates). Using information from the media items 614, an estimation about the resources (bandwidth, CPU, memory) needed to run the contained media items 614 is assigned to each configuration.

[0057] To achieve all these tasks, the MSS 504a consists of four components: MediaStructureFactories 704, MediaStructureRobots 710, a ConfigurationListManager 706 and a MediaStructureManager 702, which shall be explained in detail in the following sections.

a) MediaStructureFactories

[0058] Each time an application 302 starts to use the Adaptation Engine 402 services, it is given to a MediaStructureFactory 704 component. The MediaStructureFactory 704 implements the interface

- for constructing media structures,
- to commit media structures to the middleware framework 304,
- to remove media structures, and
- to dynamically change parts of the media structure.

[0059] Thereby, it is one of the most important components an application 302 needs for using the middleware framework 304 as it allows the application 302 to specify its adaptation possibilities.

[0060] For building a media structure, other components are needed as well. Basically, a correspondent component type exists for each aggregates type. The same applies for constraints and media items 614. In the following, all these components are called "media structure components". The MediaStructureFactory 704 allows to create instances of media structure components and to combine these "building blocks" to a media structure.

[0061] To all of these media structure components, attributes 618 can be added. Attributes 618 are specified as key-value pairs. There is a number of attributes 618 which can be "understood" by the Adaptation Engine 402. For these attributes 618, the syntax and semantics is defined by the Adaptation Engine 402. Nevertheless, the application 302 may also add parameters which the Adaptation Engine 402 does not know. This has two reasons:

- First, the Adaptation Engine 402 is extensible. Therefore, these attributes 618 may be evaluated by one of the extensions.

- Second, the attributes 618 may be given back to the application 302 after an adaptation. This is especially useful for media items 614. For example, the application 302 may store attributes 618 describing the position on the screen where the media item 614 should be shown. When the Adaptation Engine 402 decides to play the media item, it informs the application 302. In doing so, the Adaptation Engine 402 also includes the media item attributes 618 of the corresponding media item. The application 302 now knows how to position the media item 614 without having to remember all this information in a separate data structure.

[0062] Media item parameters are especially important as they play a central role in the adaptation process. If the application 302 does not specify the needed media item parameters, the middleware framework 304 tries to obtain the missing information itself. For example, this can be done by using the DESCRIBE method of the Real-Time Streaming Protocol (RTSP), or by accessing public available meta information about the media item.

[0063] The MSS 504a allows using different kinds of MediaStructureFactories 704. Different MediaStructureFactories 704 allow building media structures of different complexities. The applications 302 can request a specific kind of MediaStructureFactory 704. Some applications 302 may only need to build simple media structures, and therefore prefer to use a simple interface, whereas other applications 302 may want to create complex structures, and therefore need more complex interfaces.

[0064] As media structure components have to implement well-defined interfaces 802b and 902, it is possible to add new types of aggregates (then also new types of MediaStructureRobots 710 are needed) or new versions of existing aggregate types. This is an useful example for an application 302 which does not want to define its media structures, but wants to compute its configuration list by other means or when the configuration list is fix. In such a case, the application 302 can e.g. use an own <application> aggregate which just provides the configuration list of the application 302 without the middleware framework 304 knowing how it was created.

b) MediaStructureRobots

[0065] MediaStructureRobots 710 are components that traverse the media structure for either manipulating the structure or for retrieving information from the structure. MediaStructureRobots 710 are an implementation of the so-called "Visitor Pattern".

[0066] These MediaStructureRobots 710 consists of two parts: the basic part for traversing the media structure, and a pluggable component for performing actions on the structure. The advantage of this approach is that there can be different algorithms used for traversing the graph by different MediaStructureRobots 710. This can also dynamically be exchanged. For example, one algorithm may be more appropriate when the media structure is small, and another one when the media structure is more complex.

[0067] The MSS 504a uses different kinds of MediaStructureRobots 710: the StructureOptimizer 710a, the StructureVerifier 710b, the AdmissionControlRobot 710c, and the ConfigurationListCreator 710d, which shall now be described in detail:

- The StructureOptimizer 710a is used whenever an application 302 commits a media structure to the middleware framework 304, or when the middleware framework 304 changes the structure. Thereby, it tries to simplify the media structure. This is done by removing unnecessary aggregates (e.g. a <par> node 608 inside another <par> node 608), and combining constraints if possible.

- The StructureVerifier 710b is invoked whenever an application 302 commits a media structure to the middleware framework 304, or when the middleware framework 304 changes the structure. It checks whether the structure has inconsistencies. As the MediaStructureFactories 704 do not allow to create "illegal" structures, the basic structure has not to be verified. Instead, the StructureVerifier 710b tests whether there are semantic inconsistencies. For example, semantic inconsistencies can occur when using constraints which contradict each other.

- The AdmissionControlRobot 710c is invoked whenever an application 302 commits a media structure. It works on the committed media structure before it is added to the global structure. It finds the minimum and maximum resource requirements of the new structure and decides (based on a configurable policy) whether the new structure is accepted or rejected.

- The ConfigurationListCreator 710d is used whenever the media structure has changed. It creates a configuration list from the media structure. This MediaStructureRobot 710 is the most important one as the MSS 504a task is to compute the configuration list out of the media structure. The configuration list is the basis for adaptation decisions. The ConfigurationListCreator 710d can compute the configuration list using the whole media structure tree 600 or just a part of the tree, e.g. for one application 302. The resulting configuration list is given to the ConfigurationListManager 706.

[0068] In this context, it should be noted that the MSS 504a allows to add and/or replace MediaStructureRobots 710.

c) ConfigurationListManager

[0069] The ConfigurationListManager 706 is responsible for maintaining the configuration list. It synchronizes the access to the configuration list, which is written by the ConfigurationListCreator robot and read from the Media Control Subsystem 504b (MCS). Additionally, it has a pluggable interface 802b for adding components that manipulate the configuration list. In the following, these components are called ConfigurationListManipulators 708. As the configuration list represents the adaptation space, there are many possible usages for ConfigurationListManipulators 708. For example, they may act as filters on the configuration lists, thereby performing one of the following tasks:

- Removing configurations whose resource requirements are beyond the capabilities of the system: For example, these capabilities can be computed from the device profile of the system, which contains a description of the parts the system is composed of. This is especially interesting if future mobile terminals can dynamically change their capabilities by logically enhancing the device with other devices in its surrounding using technologies like JINI or Universal Plug-and-Play.

- Implementation of user-defined rules like "don't play audio": Such rules can easily be implemented by removing configurations which do not apply to the rule.

- Implementation of regulatory issues, e.g. if the system is used for the driver console in a car: When driving faster than a specified speed only audio is allowed. Therefore, all other configurations have to be removed until the car is driving slower again.

- Removing configurations whose resource requirements are exceeding the capabilities of the device.

- Checking the availability of the media items 614, and removing configurations where media items 614 are not accessible or not available.

- Adding new configurations, e.g. by using locally available converters or network services which manipulate multimedia streams.

[0070] As shown in Fig. 7, ConfigurationListManipulators 708 may not only act as filters but can also modify the configuration list in any way including changes to the priorities or resource consumption estimations of configurations. Additional parameters can also be added. One example would be a parameter for giving a measurement about how different the configurations are in terms of user perception (here called the "visual distance" between configurations). Two configurations that only differ in media configurations of controllable media items 614 have a shorter visual distance than two configurations which have only a subset of media items 614 in common.

[0071] If the configuration list is empty at the end of the processing, special measurements can be taken, e.g. a recomputation of the configuration list with a minimal set of ConfigurationListManipulators 708, or an addition of ConfigurationListManipulators 708, thereby increasing the number of configurations.

[0072] The ConfigurationListManager 706 informs users of the configuration list that the list has changed via corresponding events.

d) MediaStructureManager

[0073] The MediaStructureManager 702 is the controller of the MSS 504a. It coordinates the other components and implements parts of the public interface 902 of the MSS 504a (e.g. for getting a MediaStructureFactory 704).

[0074] Besides, the MediaStructureManager 702 coordinates the other components by invoking them in the right order (this applies especially for MediaStructureRobots 710). Which components are called in which order is configurable. This allows to add and/or remove functionality (for example an admission control is not always needed), or to use new enhanced types of robots. An overview of the MSS 504a is given in Fig. 7.

3. Media Control Subsystem (MCS)

[0075] The Adaptation Engine 402 implements a control loop which selects a configuration from the configuration list, and implements it by starting and configuring the contained media items 614 and informing the corresponding applications 302.

[0076] The selection is based on the current resource situation and the attributes 618 associated to each configuration.

[0077] The adaptation process is triggered by events that originate from other subsystems and the operating system 306, as can be taken from the block diagram 800 in Fig. 8:

- The MSS 504a sends events upon changes in the configuration list.
- The Media Management Subsystem 404 is supposed to periodically send status events from running media items 614. These events contain measured information of media and transmission parameters like frame rate or packet loss.
- The Local Resource Management 804 is supposed to notify problems, e.g. if the overall CPU or memory consumption exceeds certain thresholds, or if the battery is running low.
- The Local Network Management 806 is supposed to send events about the currently used access network or networks and about switches between these access networks.

[0078] The MCS 504b defines interfaces 802b to all of the subsystems 504a, 404, 804 and 806. As the subsystems 504a, 404, 804 and 806 do not implement these interfaces 802b per se, so-called wrappers 802a are needed which add the required functionality. For example, if the Media Management Subsystem 404 cannot provide the required events itself, a wrapper 802a is needed which adds this functionality, and implements the interface 802b required by the MCS 504b. Of course, wrappers 802a may be developed for different subsystems of the same kind. For example, in the case of the Media Management Subsystem 404, wrappers 802a may be developed for Quicktime and for the Java Media Framework (JMF). The MCS 504b allows using different subsystems of the same kind in parallel. For example, Quicktime and the JMF can be used in parallel if the corresponding wrappers 802a exist.

[0079] The MCS 504b manages all incoming events and decides when an adaptation decision has to be made. For this decision, the following (conflicting) aspects are taken into account:

- Agility: The user expects to see a graceful adaptation when resources get scarce. The system has therefore to react quickly on resource shortages.
- Stability: The MCS 504b has to avoid annoying the user by switching too often between different configurations that have a high perceptual difference.

[0080] It also defines the target of the adaptation by specifying the limit of the local resources the new configuration is allowed to use.

[0081] During the decision process, a new configuration is selected. This selection process has different requirements:

- It has to meet the specified target in terms of resource usage.
- It has to select the configuration which offers the best user experience (which is not mandatory the one whose resource consumption is closest to the target).
- It also has to take stability issues into account.

[0082] After a decision has been taken, the MCS 504b has to implement the decision. It has to stop media items 614 which are not part of the new configuration, start media items 614 which are in the new configuration, set the media configurations of controllable media items 614, and send events to the affected applications 302.

[0083] The implementation of the configuration has direct effects 406c on the systems generating trigger events. The MCS 504b can therefore be seen as a control loop for controlling the media items 614 based on the current resource availability.

[0084] The MCS 504b consists of three components: TriggerEventManager 908, DecisionEngine 904 and ConfigurationImplementor 906, which shall be described in detail in the following sections.

a) TriggerEventManager

[0085] The TriggerEventManager 908 receives the events from the other subsystems. For events which are generated periodically (e.g. CPU usage), additional filters can be added. An example would be to add an exponential weighted moving average filter that provides a smoothed average of the most recent observation and all past observations.

[0086] In the case of a problem (e.g. if the bandwidth of the access network has dropped significantly), the TriggerEventManager 908 will receive events from different sources. It therefore has to deal with bursts of events. Then, it has to combine the information from all events to find out the reason for the problem (e.g. whether the frame rate of a stream is going down because there is not sufficient bandwidth or not enough CPU power, or if the corresponding server 206 is overloaded). Depending on this reason, different adaptations may be required.

[0087] The TriggerEventManager 908 contains the logic to decide when an adaptation is necessary. When there are notifications about problems, an immediate reaction upon that must follow. If there are notifications that additional resources are available, the decision to do an adaptation depends on other properties as well (e.g. to achieve stability). In such a case, the decision possibly depends on the time since the last adaptation, and on the visual distance caused by it.

[0088] Additional to the events, the TriggerEventManager 908 can actively ask for data. (The methods therefore are also designed in the interfaces 910.) Using both methods, the TriggerEventManager 908 has a good knowledge about the current resource consumption, and - in case of a problem - about the amount of missing resources. It can therefore calculate a target in terms of the resource usage for the Adaptation Engine 402. For the network resources, the target can include overall information as well as targets for specific network routes. For example, the traffic to note X should not exceed the available bandwidth B_x.

[0089] Since the implementation of an adaptation takes some time (as media streams may have to be started), the TriggerEventManager 908 assures that there is a configurable period of time after an adaptation where no new adaptation can be performed, and where all incoming events are ignored. In the following, this period of time shall be called the "protection period".

[0090] If the sender of trigger events are only capable of sending events in case of problems, the TriggerEventManager 908 may use one of the following three approaches:

- Making of own measurements: For example, this can be done with the available bandwidth for a specific route - a value that is normally not notified if it is increasing.

- Probing: In this case, the TriggerEventManager 908 triggers the DecisionEngine 904 periodically or based on an heuristic model to switch to the configuration with the next higher resource consumption. Thereby, the probing behavior must be configurable by the user.

- Allowing the user to specify when to adapt and to which configuration, e.g. via a Graphical User Interface (GUI): In this case, the Adaptation Engine 402 runs in an "half automatic" mode, where it is automatically reacting upon problems (resource shortages) but not when the situation gets better.

b) DecisionEngine

[0091] The TriggerEventManager 908 triggers the DecisionEngine 904. Moreover, the TriggerEventManager 908 provides a target in terms of the resource consumption for the Adaptation Engine 402. The DecisionEngine 904 gets the actual configuration list from the MSS 504a, and maintains knowledge about the current active configuration.

[0092] Having all these information, the DecisionEngine 904 selects the configuration to switch to. This decision can be made as follows:

- Each configuration contains attributes 618 describing its estimated resource consumption. The resource consumption is the primary set of parameters for selecting a configuration. Configurations that have a lower resource consumption than the targets are called "possible configurations". If there is no such configuration, the configuration which is the "closest" to the target is selected. In this case, the selection process is finished. To find out which configuration is the "closest" to the target, the DecisionEngine 904 contains a configurable metric for the "distance" between two sets of resource consumption parameters. It is configurable to allow the specification of different weights for each resource type. In some installations, bandwidth may be the most important parameter, in others (e.g. for small PDAs) it may be the CPU load. If no configuration can be found, events to the applications 302 are thrown. The applications 302 may have the possibility to change their media structure upon such an event. Which applications 302 are notified depends on the priority of the different applications 302. Alternatively or additionally to sending events, the DecisionEngine 904 can inform the user about the problem, who may then close media items 614 or applications 302. The behavior of the DecisionEngine 904 in such a case is configurable.

- If there is more than one possible configuration, the new configuration is selected by taking two additional configuration parameters into account: the priority of the configuration and the visual distance compared to the current configuration. These two parameters are aggregated to one by mapping the visual distance to a value that can be subtracted from the priority. The mapping function takes the time since the last configuration change into account. The longer the time since the last configuration change is, the smaller gets the value. The mapping function is configurable to adapt the behavior to the preferences of the users. For some users, stability will play a more important role than for others.

[0093] After having selected a configuration, the configuration information is passed to the ConfigurationImplementor 906 which implements the new configuration.
[0094] The DecisionEngine 904 can principally handle any kind of resources as it just matches the resource information in the configuration list with the target specified by the TriggerEventManager 908. This can be done by just comparing the names of the resources. From the point of view of the DecisionEngine 904, resources can therefore easily be added. Unfortunately, this is not so easy for the other components. To add a new resource means that the MSS 504a has to provide a resource estimation for the new resource, and the TriggerEventManager 908 has to get information about the correspondent resource situation to be able to specify a target. All these components are designed in such a way that the handling for additional resources can be added, but it involves some effort. To avoid this work, the Adaptation Engine 402 already contains logic for handling lots of different resources. Which of them are actually used depends on the availability of the resource information. For example, on a notebook the battery status is taken into account, which is not the case on a desktop computer. Whether a resource is used in the adaptation process depends upon whether the MSS 504a can provide an estimation of the resource usage for each configuration, and if the TriggerEventManager 908 can measure this resource usage. If both is true, the resource is taken into account.
[0095] Sometimes, users want total control about what is happening in their system. The DecisionEngine 904 therefore allows a "manual" mode, in which different configurations are presented to the user, and the user decides when to adapt and which configurations to use. In this case, no trigger events are automatically processed by the TriggerEventManager 908. Furthermore, the information contained in the events is visualized to help the user making decisions. In the manual and half-automatic mode, it makes also sense to allow the user to make not only a terminal-wide decision but to decide e.g. for individual applications 302. To make this possible, the ConfigurationListManager 706 has to provide the configuration lists for the <app> aggregates in the media structure. These can be visualized to the user per application, and the selected configuration can be implemented without changing the configurations of the other applications 302. The same approach can be performed with any other aggregate level in the media structure.

c) ConfigurationImplementor

[0096] The ConfigurationImplementor 906 gets as input from the DecisionEngine 904 the new configuration, and maintains information about the current configurations.
[0097] Its task is to implement the new configuration. This involves the following steps:

- stopping all media items 614 that area part of the current configuration but not part of the new configuration,
- starting all media items 614 that are not part of the current configuration but part of the new configuration,
- setting off the media configuration of the new media items 614 if they are controllable,
- setting the media configurations of all controllable media items 614 that are part of the current and new configuration (if different from the current media configuration).

[0098] The ConfigurationImplementor 906 has to inform the corresponding application 302 about started or stopped media items 614. This is done by sending so-called adaptation events to the application 302, which - in case of a new media item 614 - also contain a reference to a media player object which can be used by the application 302 to play the media item 614 and to control it (starts, stop, pause, etc.). These events also include any parameters specified by the application 302 when defining its media structure but not used by the Adaptation Engine 402.

[0099] The ConfigurationImplementor 906 also maintains for each application 302 a control component (media controller) that allows the application 302 to simultaneously control all continuous media items 614 belonging to it (start, stop, pause, etc.). New continuous media items 614 are added to the corresponding media controller, stopped media items 614 are removed from it.

[0100] The problem when switching between different continuous media items 614 is at what position (media time) the new continuous media item 614 should start. The Adaptation Engine 402 supports different possibilities that can be specified for each continuous media item.

[0101] The possibilities are:

- Starting always from the beginning (media time = 0): This is the default for live streams, which do not have a specified duration.

- Starting always from a predefined time.

- Starting always from the last media time of the media item: In this case, if the media item 614 is started, stopped and then restarted again, it continues as the same time it has stopped.

- Using the time base of the corresponding <switch> aggregate 612: If there is a change between two media items 614 belonging to the same <switch> aggregate 612, they can start based on the time the first media item 614 in the <switch> aggregate 612 was started. This is called the "switch time base". Thus, a media item 614 starts at media time, which given by the following equation:

$$\text{media time} = \text{current time} - \text{switch time base}.$$

- Using the time base of the corresponding <app> aggregate 606: This is the same as already described for the <switch> aggregate 612. The media item 614 starts at a time based on the "app time base". Thereby, the media time is given by the following equation:

$$\text{media time} = \text{current time} - \text{app time base}.$$

- Using the "switch time base" or the "app time base" without taking times into account where there is no continuous media item 614 played in the corresponding <switch> aggregate 612 or <app> aggregate 606: An example for this would be a <switch> aggregate 612 with three media items 614: video_A, video_B and image. The first adaptation changes from video_A to image, and the second from image to video_B. Then, video_B starts at the last media time of video_A and not (as it would be the case with just using the switch time base) at the last media time of video_A plus the time the image was displayed.

[0102] In theory, additional time bases could be added (e.g. for each aggregate), but the described ones are the ones which make most sense for the application 302. In any case, the application 302 has the possibility to change the media time before actually starting the media player contained in the adaptation event.

[0103] The ConfigurationImplementor 906 also contains the logic to use network reservation mechanisms if available. It may either use the mechanism itself, or use the interfaces 802b provided by other subsystems like the Media Management Subsystem 404 for doing the reservation. If the reservation fails, it sends a trigger event to the TriggerEventManager 908 containing the result of the reservation (which means the reason why it failed). The TriggerEventManager 908 then starts a new adaptation process with an adapted target. An overview of the MCS 504b is given in Fig. 9.

[0104] The Adaptation Engine 402 contains some components that are not part of the MSS 504a and the MCS 504b, but used by different components of the Adaptation Engine 402. Such common components offer services needed by a set of other components. Especially, two components are important: the adaptation context unit 912 and the Graphical User Interface (GUI) Manager 914.

- Adaptation context unit 912: Although the middleware framework 304 is targeted to be made of different components which are highly independent from each other, the components have to share a lot of information. Such information is stored in the adaptation context unit 912. It can be accessed by the components to store and retrieve data.

- GUI Manager 914: The Adaptation Engine 402 comprises a Graphical User Interface (GUI) which offers a comprehensive functionality, e.g. for providing user interaction and for monitoring the adaptation process. To fulfill this task, the GUI needs to have access to different components to retrieve data, and to manipulate the components. The GUI Manager 914 is a central component which offers an API with all the functionality the Adaptation Engine 402 offers for use by a GUI. Using this API, different GUIs can be developed. Inexperienced users may prefer a simple GUI with only little options, whereas experienced users may want to have detailed information about the adaptation process and more options to influence the adaptation process.

[0105] In some scenarios, as depicted in Fig. 10, the client system does not consist of only one client terminal 204 but of different client terminals 204 communicating over a Personal Area Network (PAN). For example, this is the case in car (or train) scenarios where different devices may use the car infrastructure for being connected to the Internet 202. In such scenarios some resources (e.g. network bandwidth) are not controlled by the client terminals 204 but at a central place accessible to all client terminals 204. These entities are called "Central Resource Controllers" 1002 (CRC). Thereby, the Adaptation Engine 402 has to work closely together with the CRCs 1002 to make useful adaptation decisions.

[0106] Furthermore, in such a scenario the Adaptation Engine 402 has to be distributed on different nodes. In the following, it shall be assumed that there is only one CRC 1002, and a set of client terminals 204 using this CRC 1002. Thereby, a MSS 504a runs on each client terminal 204 to let the applications 302 define their media structure. The media structure is sent to the local MCS 504b which, in case of an adaptation, selects the set of possible configurations based upon the resources managed by the client terminal 204. The MCS 504b also modifies the list of possible configurations by aggregating the priority and visual distance. The resulting configuration list is sent to the CRC 1002. The CRC 1002 queries all other client terminals 204 for similar configuration lists and combines these into a global list of possible configurations by still maintaining the information about the original configurations. The MCS 504b of the CRC 1002 selects a configuration by taking the resource it is managing and the priority values of the configurations into account. Next, it looks up the original configurations that formed the selected configuration (during aggregation). Finally, it informs the correspondent client terminals 204 (to be more specific the ConfigurationImplementors 906 of the client terminals 204) about which configuration they have to implement.

[0107] In the described case, the adaptation is triggered by the TriggerEventManagers 908 of the client terminals 204. If the CRC 1002 can itself start the adaptation process, it does this by querying all client terminals 204 for their list of possible configurations.

[0108] In this case, the actual decision process is distributed between the client terminals 204 and the CRC 1002. If there is more than one CRC 1002, the decision is further distributed. This is done by introducing an hierarchy between the CRCs 1002, as depicted in Fig. 11. The first CRC 1002 in the hierarchy computes the global list of possible configurations. It removes all configurations that are not possible due to the availability of the locally managed resources. The reduced list is then passed to the next CRC 1002 in the hierarchy which again removes the configurations not matching with its resource target. The last CRC 1002 in the hierarchy makes the final decision and informs the first CRC 1002 which in turn informs the client terminals 204 about the configurations to implement. If the information about how the configurations were assembled from the device configurations is passed along with the configuration list, the last CRC 1002 can directly inform the client terminals 204.

[0109] There are scenarios in which the user wants to switch to another client terminal 204 and wants to start the new client terminal 204 in exactly the same state as he left the original one. The Adaptation Engine 402 supports this if there is an Adaptation Engine 402 installed on both client terminals 204.

[0110] The Adaptation Engine 402 offers an API to programmatically retrieve and restore the internal status of the Adaptation Engine 402 for a specific user, for a specific application 302 or for the whole terminal. The status information can inbetween be transferred to the other client terminal 204. How the status information is transferred to the other client terminal 204 is out of the scope of the Adaptation Engine 402, as in such a scenario not only the status of the Adaptation Engine 402 has to be transferred but also the status of the application 302 and other subsystems. Therefore, an entity external to the Adaptation Engine 402 is needed to collect the different status information, to transfer it to the other client terminal 204, and to restore it again.

[0111] The status of the Adaptation Engine 402 comprises the media structure, the configuration list, the current configuration, and for each continuous media item 614 the current media time.

[0112] Using this information, the Adaptation Engine 402 on the target client terminal 204 can restore the original status. The first thing after restoring the information is to perform an adaptation as the target client terminal 204 may have different capabilities and a different resource availability than the original client terminal 204.

[0113] An example how the described middleware framework 304 according to the underlying invention could be used for multimedia retrieval applications 200 (in contrast to conversational and messaging applications 302) is to use a multimedia document browser 1202 on top of the proposed middleware framework 304 as depicted in Fig. 12. A multimedia document browser 1202 is very similar to standard Web browsers 1202, but presents documents containing multimedia content to the user. The document describes the layout of the multimedia presentation and contains references to the media items 614 which are separate from the document. By using a document language which allows specifying the media structure already in the document, the multimedia document browser 1202 could easily use the middleware framework 304. Furthermore, using the multimedia document browser 1202 makes it easy to develop adaptive multimedia applications 302 by writing such documents.

[0114] The significance of the symbols designated with reference signs as depicted in Figs. 1 to 12 can be taken from the appended table of reference signs.

[0115] In the following, the main advantageous differences between the preferred embodiment of the underlying invention and the state of the art shall be emphasized.

[0116] The described middleware framework 304 is unique in its functionality. It provides a middleware framework 304 for controlling the adaptation process for distributed multimedia applications 302. It especially takes the dynamic nature of wireless networks into account, and is therefore very well suited for mobile clients 204. The targeted application type is general enough to support a large number of applications 302 - and these applications 302 are the ones that definitely profit from, and need adaptation mechanisms in a mobile scenario. On the other hand, the targeted application type is special enough to provide rich adaptation support by incorporating domain knowledge in the Adaptation Engine 402.

[0117] For the targeted distributed multimedia applications 302, the middleware framework 304 offers a comprehensive support. In contrast to other frameworks, it uses domain knowledge and allows the applications 302 to specify their adaptation possibilities, thus taking detailed domain knowledge and adaptation knowledge into account for the adaptation process. Especially, properties and problems of streaming media are taken into account by the middleware framework 304.

[0118] In contrast to some other systems that support an adaptation based on static parameters, the middleware framework 304 supports a dynamic adaptation to change resource availability, user preferences and device constraints.

[0119] The middleware framework 304 allows applications 302 to get different levels of support from the middleware framework 304. Applications 302 wanting more control over their media can achieve this by giving less medium structure information to the middleware framework 304. The more information the application 302 provides, the more support it gets from the middleware framework 304. Therefore, various types of applications 302 can benefit from the proposed middleware framework 304 by using exactly the amount of support they need for accomplishing their tasks.

[0120] The middleware framework 304 includes very flexible mechanisms to enable the applications 302 to specify their adaptation knowledge. The provided mechanisms are general enough to support a wide range of applications 302 and do not restrict the applications 302 in their functionality.

[0121] Furthermore, the middleware framework 304 can very flexibly be configured and tailored to the user and application 302 needs. It can also easily be enhanced, and was designed to be useful in different scenarios ranging from small devices to client systems that consist of a set of distributed client terminals 204. The middleware framework 304 can even run in a distributed set-up to allow adaptation decisions in more complex client scenarios 1000.

[0122] The middleware framework 304 also takes the user requirements and preferences into account. It allows users a detailed specification of the control he/she wants to have over the adaptation process. With the concept of the so-called "visual distance" and various configuration options, the user can tailor the system to behave in a way which suits him best.

[0123] The middleware framework 304 is flexible enough to work together with a plurality of available systems. For example, it can use and work together with different existing Media Management Subsystems 404.

[0124] The middleware framework 304 is a client-only architecture that uses standard protocols and data types. It can therefore use the existing multimedia infrastructure, e.g. media servers 1206 and also available content. Thereby, no changes have to be made on the server or network side to use the middleware framework 304.

[0125] In Table 1, the underlying invention is compared with already known technologies according to the state of the art.

Table 1:

Comparison to related work	
Technologies	Comment
Java Media Framework (JMF),	These technologies are not framework-supporting adaptations but media management systems that provide components for the ac-

Table 1: (continued)

Comparison to related work	
Technologies	Comment
DirectShow, MPEG-21, Quicktime	tual media handling. By contrast, the proposed middleware framework 304 can be seen as an extension to such media management systems adding adaptation support. Some of the mentioned frameworks according to the state of the art foresee simple methods for adapting single media streams (e.g. by supporting an easy creation of different qualities of a video), whereas the middleware framework 304 proposed by the underlying invention can use all these methods. Furthermore, one advantage of the proposed middleware framework 304 is that it is not bound to a specific media management system.
RealPlayer / SureStream Technology	The RealPlayer implements SMIL. Additionally, it allows using the SureStream technology that allows a single media item to dynamically adapt to different resource situations. An adaptation is then done in two steps: - First, an alternative is selected using the information in the SMIL document.
	- Second if the chosen stream is a SureStream, it dynamically adapts to the resource situation. The difference to the proposed middleware framework 304 according to the underlying invention is that there is no coordination between different running SureStreams. They are operating independently of each other, and there is a never a re-evaluation of the other alternative specified in the SMIL document. Moreover, SureStreams only adapt to bandwidth fluctuations, and not to other resources like CPU and memory.
TIEMPO	The main focus of TIEMPO was the development of a temporal model for multimedia presentations. The TIEMPO document language and presentation system also includes adaptation

Table 1: (continued)

Comparison to related work	
Technologies	Comment
	mechanisms. Especially, the dynamic scheduling of media items based on a bandwidth estimation mechanism is an adaptation mechanism that is not covered in the proposed middleware framework 304 according to the underlying invention. In contrast to the proposed middleware framework 304, the TIEMPO presentation system is not a middleware framework 304 but an integrated multimedia player. Therefore, adaptation decisions are only possible for the currently executed documents. In this context, it should be noted that the TIEMPO presentation system does not support multimedia applications 302 that are not document-based.
	TIEMPO is also not targeted to wireless environments. Furthermore, TIEMPO's adaptation mechanism reacts slowly on bandwidth changes. The bandwidth estimation mechanisms used probably do not work correctly in a dynamic environment. Moreover, TIEMPO does not take actual measurements of the media items (e.g. packet loss) into account for the adaptation process.

Table 2:

Depicted Features and their Corresponding Reference Signs	
No.	Feature
100	example of an adaptive application for adapting two distributed multimedia application 302 scenarios with different bandwidths and different Quality of Service (QoS) in heterogeneous communication networks
102	window showing an example of a broadband distributed multimedia application scenario
104	window showing an example of a narrowband distributed multimedia application scenario
200	example for a multimedia retrieval application with distributed servers 206, 208 and one client 204 connected to the Internet 202
202	Internet
204	client connected to the Internet 202
206	multimedia (streaming) server connected to the Internet 202
208	World Wide Web (WWW) server connected to the Internet 202
300	client architecture comprising a middleware framework 304
302	distributed multimedia application running on a client 204
304	middleware framework 304

Table 2: (continued)

Depicted Features and their Corresponding Reference Signs	
No.	Feature
306	operating system
308	hardware components
400	overall architecture of the middleware framework 304 comprising an Adaptation Engine 402 and a Media Management Subsystem 404
402	Adaptation Engine
404	Media Management Subsystem
406a	demultiplexer
406b	codec
406c	effects
406d	multiplexer
406e	renderer
500	detailed block diagram of the overall architecture 400 of the middleware framework 304
502	Multimedia Application Manager (MAM)
504a	Media Structure Subsystem (MSS)
504b	Media Control Subsystem (MCS)
600	example of a media structure tree
602	<root> node of the media structure tree 600
604	<user> node of the media structure tree 600
606	<app> (application) node of the media structure tree 600
608	<par> (parallel mode) node of the media structure tree 600
609	<seq> (sequential mode) node of the media structure tree 600
610	<sync> (synchronization) node of the media structure tree 600
612	<switch> node of the media structure tree 600
614	<mi> (media item) node of the media structure tree 600
616	<constraint> node of the media structure tree 600
618	attributes belonging to the nodes of the media structure tree 600
700	overview of an architecture comprising a Media Structure Subsystem 504a (MSS)
702	MediaStructureManager
704	MediaStructureFactory
706	ConfigurationListManager
708	ConfigurationListManipulator
710	MediaStructureRobot
710a	StructureOptimizer
710b	StructureVerifier
710c	AdmissionControlRobot
710d	ConfigurationListCreator

Table 2: (continued)

Depicted Features and their Corresponding Reference Signs	
No.	Feature
800	block diagram showing the interfaces 802b of the Media Control Subsystem 504b (MCS) to other subsystems 404, 504a, 804 and 806
802a	wrapper connected to the Media Structure Subsystem 504a
802b	interface between the Media Control Subsystem 504b and the wrapper 802a
804	Local Resource Management (e.g. operating system 306)
806	Local Network Management (e.g. operating system 306)
900	overview of the Media Control Subsystem 504b (MCS)
902	interface between the Media Structure Subsystem 504a and the DecisionEngine 904
904	DecisionEngine
906	ConfigurationImplementor
908	TriggerEventManager
910	interface between the TriggerEventManager 908, the Media Management Subsystem 404 and the Local Network Management components 804 and 806, respectively
912	adaptation context unit
914	Graphical User Interface (GUI) Manager
1000	example of a client 204 consisting of different terminals and a central communication management 1002, called Central Resource Controller (CRC)
1002	Central Resource Controller (CRC)
1100	example of distributed client terminals 204 with two Central Resource Controllers 1002 (CRCs)
1200	example of an application with a multimedia document browser 1202 using the middleware framework 304
1202	multimedia document browser
1204	application server
1206	media server

Table 3:

Used Terminology	
Term	Definition
Adaptation	Mechanisms altering the application behavior to compensate any fluctuations and/or even dramatic changes in data transmission and/or processing quality based on user requirements specifying said quality. These mechanisms can be included in the application or provided by some external entity (e.g. middleware).
Application	A computer program with an user interface. The underlying invention, however, addresses with this term also computer programs acting as service providers, as e.g. a Video-on-Demand server. This kind of applications usually does not locally feature any interaction with users, except for a system administration interface.
Component	A pre-fabricated customizable software unit featuring well-defined interfaces. Examples of components are Java-Beans, DCOM objects or CORBA objects.
Connection	Semi-permanent relationship between two nodes in a network. Within the context of said relationship, data can be exchanged in an exclusive and controlled manner.

Table 3: (continued)

Used Terminology	
Term	Definition
Middleware framework	A set of correlated software units embodying an abstract design for solutions to a number of related problems.
User mobility	The ability of a user to access telecommunications services from different terminals and different networks, and the capability of the network to identify and authorize that user.
Terminal mobility	The ability of the terminal to access telecommunications services from different locations and, while in motion, the capability of the network to identify and locate the terminal.
Media item	A media item is an abstraction for different kinds of media. It can be further distinguished between discrete media items (e.g. images) and continuous media items that represent time-based media (e.g. audio or movie clips). Sometimes, continuous media items can be controlled concerning their quality and resource consumption. In the context of the underlying invention, these media items 614 are called "controllable media items".
Media structure	A data structure that includes media items and relationships between media items.
Multimedia application	Multimedia applications or more general multimedia systems can be defined as follows (according to [Steinmetz93]): "A multimedia system is characterized by a computer-based, integrated production, manipulation, representation, storage and communication of independent information, coded in at least one continuous (time-dependent) and one discrete (time-independent) medium."
Process	A part of a running software program or other computing operation that performs a single task.
Quality of Service (QoS)	"The collective effect of service performance which determines the degree of satisfaction of a user of the service." (Definition from the ITU-T [former CCITT] recommendation E.800)
Stream / Flow	The sequence of information being transmitted from the source end point to the destination end point of a given connection.

Table 4:

Used Abbreviations	
Abbr.	Explanation
API	Application Programming Interface
APP	Application

Table 4: (continued)

Used Abbreviations	
Abbr.	Explanation
CPU	Central Processing Unit
CRC	Central Resource Controller
GPRS	General Packet Radio Service
GUI	Graphical User Interface
IETF	Internet Engineering Task Force
IP	Internet Protocol
JMF	Java Media Framework
MAC	Medium Access Control
MAM	Multimedia Application Manager
MCS	Media Control Subsystem
MI	Media item
MSS	Media Structure Subsystem
OS	Operating System
PDA	Personal Digital Assistant
QoS	Quality of Service
RTP	Real-Time Protocol
RTSP	Real-Time Streaming Protocol
SMIL	Synchronized Multimedia Integration Language
WLAN	WaveLAN
XML	Extensible Markup Language

Table 5:

Publications	
Abbr.	Publication
[Chang2000]	F. Chang, V. Karamcheti: "Automatic Configuration and Run-time Adaptation of Distributed Applications", Ninth IEEE Symposium on High Performance Distributed Computing, August 2000.
[Chang2001]	F. Chang, V. Karamcheti: " A Framework for Automatic Adaptation of Tunable Distributed Applications", Cluster Computing: The Journal of Networks, Software and Applications, Vol. 4, No. 1, 2001.
[Diot98]	C. Diot, C. Huitema, T. Turletti: "Multimedia Applications Should Be Adaptive", HPCS'95 Workshop, 1998. ftp://www.inria.fr/rodeo/diot/ncahpcs.ps.gz http://citeseer.nj.nec.com/diot98multimedia. html
[Gamma95]	E. Gamma, R. Helm, R. Johnson, J. Vlissides: "Design Patterns, Elements of Reusable Object-Oriented Software", Addison-Wesley 1995, ISBN 0-201-63361-2.
[Gecsei97]	J. Gecsei: "Adaptation in Distributed Multimedia Systems", IEEE Multimedia, Vol. 4 2, April-June 1997, Pages 58-66.
[Li2000]	B. Li: "Agilos: A Middleware Control Architecture for Application-Aware Quality-of-Service Adaptations", PhD Dissertation, Department of Computer Science, University of Illinois at Urbana-Champaign, May 2000.

Table 5: (continued)

Publications	
Abbr.	Publication
[Noble2000]	B. Noble: "System Support for Mobile, Adaptive Applications", IEEE Personal Communications, Vol. 7, No. 1, February 2000.
[Noble97]	B. Noble, M. Satyanarayanan, D. Narayanan, J.E. Tilton, J. Flinn, K. Walker: "Agile Application-Aware Adaptation for Mobility", Proceedings of the 16th ACM Symposium on Operating System Principles, October 1997, St. Malo, France.
[Satyanarayanan94]	M. Satyanarayanan, B. Noble, P. Kumar, M. Price: "Application-Aware Adaptation for Mobile Computing", Proceedings of the 6th ACM SIGOPS European Workshop, September 1994, Dagstuhl, Germany.
[Satyanarayanan96]	M. Satyanarayanan: "Fundamental Challenges in Mobile Computing", 15th ACM Symposium on Principles of Distributed Computing, May 1996, Philadelphia, PA. Revised version appeared as: "Mobile Computing: Where's the Tofu?", Proceedings of the ACM Sigmobile, April 1997, Vol. 1, No. 1.
[Shulz98]	H. Schulzrinne, A. Rao, R. Lanphier: "Real-Time Streaming Protocol (RTSP)", IETF Request for Comments: 2326, April 1998.
[SMIL98]	"Synchronized Multimedia Integration Language (SMIL) 1.0 Specification", W3C Recommendation, 15th June 1998.
[Steinmetz93]	R. Steinmetz, "Multimedia-Technologie: Einführungen und Grundlagen", Springer-Verlag, 1993.
[SureStream]	Real Networks Whitepaper: "SureStream™ - Delivering Superior Quality and Reliability" http://www.realnetworks.com/devzone/library/ whitepapers/surestrm.html [Accessed: 26th September 2001].
[Vogel95]	A. Vogel, B. Kerhervé, G. v. Bochmann, J. Gecsei: "Distributed Multimedia and QoS: A Survey", IEEE MultiMedia 2(2): 10-19, 1995.
[Wahl95]	T. Wahl, S. Wirag, K. Rothermel: "TIEMPO: Temporal Modeling and Authoring of Interactive Multimedia", in: IEEE 2nd. Intl. Conference on Multimedia Computing and Systems, Washington D.C., USA, May 1995, pp. 274-277.
[Wirag97]	S. Wirag, R. Steinmetz, L.C. Wolf (Eds.): "Modeling of Adaptable Multimedia Documents", Interactive Distributed Multimedia Systems and Telecommunication Services, 4th International Workshop, IDMS'97, Darmstadt, Germany, September 1997, pp. 420-429.

Claims

1. A middleware framework system implemented on a client (204) targeted to support distributed multimedia applications (302) in heterogeneous communication networks to dynamically adapt to varying resource availability, comprising:

   - at least one Media Management Subsystem (404) for managing multimedia data to periodically send status events from running media items (614), which contain measured information of media and transmission parameters,
   - at least one Adaptation Engine (402) for controlling the adaptation of said distributed multimedia applications (302) to the current resource availability,

   characterized in that
   said middleware framework system is designed to (304) allow each running distributed multimedia application (302) to specify the media it wants to use and the relationships between these media, calculates the adaptation possibilities and controls the adaptation process.

2. A middleware framework system according to claim 1,

characterized in that the Adaptation Engine (402) comprises:

- at least one Multimedia Application Manager (502), which is responsible for offering the Application Programming Interface of the Adaptation Engine (402) to the distributed multimedia applications (302),
- at least one Media Structure Subsystem (504a), which gets information from the distributed multimedia applications (302), and aggregates this information to a list of possible configurations, and
- at least one Media Control Subsystem (504b) implementing a control loop which dynamically selects a configuration based upon the current resource availability, implements it and notifies the affected distributed multimedia applications (302).

3. A middleware framework system according to claim 2,
characterized in that
said Multimedia Application Manager (502) connects the contained subsystems during the start-up of a distributed multimedia applications (302), and calls the corresponding methods to decouple and deallocate the other subsystems during the shut-down of said distributed multimedia applications (302).

4. A middleware framework system according to anyone of the claims 2 and 3,
characterized in that
the Media Structure Subsystem (504a) comprises:

- at least one MediaStructureFactory (704), which implements the interface for constructing media structures, to commit media structures to the middleware framework system (304), to remove media structures, and to dynamically change parts of the media structure,
- at least one MediaStructureRobot (710), which traverses the media structure for either manipulating the structure or for retrieving information from the structure,
- at least one ConfigurationListManager (706), which is responsible for synchronizing the access to the configuration list read from the Media Control Subsystem (504b), and adding ConfigurationListManipulators (708) that manipulate the configuration list, and
- at least one MediaStructureManager (702), which creates the MediaStructureFactories (704) and controls the MediaStructureRobots (710) and the ConfigurationListManager (706).

5. A middleware framework system according to anyone of the claims 2 to 4,
characterized in that
the Media Structure Subsystem (504a) is designed to perform the following actions:

- management of a media structure tree (600) consisting of media elements and media items (614) both with associated attributes (618), which change dynamically as the distributed multimedia applications (302) add or remove media structure information,
- calculation of a list of alternative configurations describing different choices that result from the evaluation of the media structure, which is needed by the Media Control Subsystem (504b) to make adaptation decisions,
- association of a list of attributes (618) to each configuration by using the priorities from the aggregates,
- assignment of an estimation about the resources needed to run the contained media items (614) to each configuration by using information from the media items (614).

6. A middleware framework system according to anyone of the claims 4 to 5,
characterized in that
the Media Structure Subsystem (504a) is designed to use different kinds of MediaStructureRobots (710):

- at least one StructureOptimizer (710a), which is used to simplify the media structure whenever a distributed multimedia application (302) commits a media structure to the middleware framework system (304), or when the middleware framework system (304) changes the media structure,
- at least one StructureVerifier (710b), which is applied to checks whether the media structure has semantic inconsistencies, and invoked whenever a distributed multimedia application (302) commits a media structure to the middleware framework (304), or when the middleware framework (304) changes the media structure,
- at least one AdmissionControlRobot (710c), which is employed to find the minimum and maximum resource requirements of a new media structure, and to decide whether the new media structure is accepted or rejected, and which is invoked whenever a distributed multimedia application (302) commits a media structure,
- at least one ConfigurationListCreator (710d), which is applied to create a configuration list from the media

structure given to the ConfigurationListManager (706) in order to make adaptation decisions, and invoked whenever the media structure has changed.

7. A middleware framework system according to anyone of the claims 4 to 6,
characterized in that
the Media Structure Subsystem (504a) is designed to allow an addition and/or replacement of MediaStructureRobots (710).

8. A middleware framework system according to anyone of the claims 4 to 7,
characterized in that
the ConfigurationListManipulators (708) can be used as filters on the configuration lists, thereby performing one of the following tasks:

- removing configurations whose resource requirements are beyond the capabilities of the applied distributed multimedia application (302),
- executing user-defined rules by removing configurations which do not apply to these rules,
- executing regulatory issues in exceptional cases by removing other configurations,
- removing configurations whose resource requirements exceed the capabilities of the applied distributed multimedia application (302),
- checking the availability of the media items (614) and removing configurations where media items (614) are not accessible or not available, and
- adding new configurations by using locally available converters or network services which manipulate multimedia streams.

9. A middleware framework system according to anyone of the claims 4 to 8,
characterized in that
the ConfigurationListManipulators (708) can be used to modify the configuration list to change the priorities or resource consumption estimations of configurations.

10. A middleware framework system according to anyone of the claims 4 to 9,
characterized in that
the ConfigurationListManager (706) informs users of the configuration list via corresponding events if the configuration list has been modified.

11. A middleware framework system according to anyone of the claims 4 to 10,
characterized in that
the MediaStructureManager (702) invokes the MediaStructureRobots (710) in a predefined order.

12. A middleware framework system according to anyone of the claims 2 to 11,
wherein the Media Control Subsystem (504b) defines interfaces (802b) to the Media Structure Subsystem (504a), the Media Management Subsystem (404), at least one Local Resource Management (804) and at least one Local Network Management (806) by using wrappers (802a) adding the required functionality,
characterized in that
the Media Control Subsystem (504b) is designed to

- manage all incoming events that originate from the Media Structure Subsystem (504a), the Media Management Subsystem (404), the Local Resource Management (804) and the Local Network Management (806),
- decide when an adaptation decision has to be made,
- define the target of the adaptation by specifying the limit of the local resources a new configuration is allowed to use,
- select a new configuration, and
- implement the decision by stopping media items (614) which are not part of the new configuration, starting media items (614) which are part of the new configuration, setting the media configurations of controllable media items (614), and sending events to the affected distributed multimedia applications (302).

13. A middleware framework system according to anyone of the claims 2 to 12,
characterized in that the Media Control Subsystem (504b) comprises:

- at least one TriggerEventManager (908), which receives said events from the Media Management Subsystem (404), the Local Resource Management (804) and the Local Network Management (806), finds out the reason for problems indicated by said events, and provides a target in terms of the resource consumption for the Adaptation Engine (402) depending on that reason,
- at least one DecisionEngine (904) triggered by the TriggerEventManager (908), which gets the actual configuration list from the Media Structure Subsystem (504a), maintains knowledge about the current active configuration, and selects a new configuration to switch to, and
- at least one ConfigurationImplementor (906), which gets as input from the DecisionEngine (904) the new configuration, maintains information about the current configurations, and implements said new configuration.

14. A middleware framework system according to anyone of the claims 2 to 13,
characterized in that
the TriggerEventManager (908) contains a logic to decide when an adaptation is necessary depending on the availability of resources and stability issues.

15. A middleware framework system according to anyone of the claims 2 to 14,
characterized in that
after the implementation of an adaptation the TriggerEventManager (908) assures that there is a configurable period of time where no new adaptation can be performed, and where all incoming events are ignored.

16. A middleware framework system according to anyone of the claims 2 to 15,
characterized in that
the TriggerEventManager (908) is able to make own measurements using the available bandwidth for a specific route, triggers the DecisionEngine (904) periodically or based on an heuristic model to switch to the configuration with the next higher resource consumption, or allows the user to specify when to adapt and to which configuration if the Media Management Subsystem (404), the Local Resource Management (804) and the Local Network Management (806) are only capable of sending events in case of problems.

17. A middleware framework system according to anyone of the claims 2 to 16,
characterized in that
the DecisionEngine (904) allows a manual mode, in which different configurations and the information contained in the events are presented to the user who decides when to adapt and which configurations to use.

18. A middleware framework system according to anyone of the claims 2 to 17,
characterized in that
the implementation of a new configuration by the ConfigurationImplementor (906) involves the following steps:

- stopping all media items (614) that are a part of the current configuration but not part of the new configuration,
- starting all media items (614) that are not part of the current configuration but part of the new configuration,
- setting off the media configuration of the new media items (614) if they are controllable,
- setting the media configurations of all controllable media items (614) that are part of the current and new configuration if they are different from the current media configuration.

19. A middleware framework system according to anyone of the claims 2 to 18,
characterized in that
the ConfigurationImplementor (906) informs the corresponding distributed multimedia application (302) about started or stopped media items (614) by sending adaptation events to the respective distributed multimedia application (302) which - in case of a new media item (614) - also contain a reference to a media player object which can be used by the distributed multimedia application (302) to play said media item (614) and to control it.

20. A middleware framework system according to anyone of the claims 2 to 19,
characterized in that
the ConfigurationImplementor (906) maintains a media controller for each distributed multimedia application (302) that allows the distributed multimedia application (302) to simultaneously control all continuous media items (614) belonging to it by adding new continuous media items (614) to the corresponding media controller, and removing stopped media items (614) from it.

21. A middleware framework system according to anyone of the claims 2 to 20,

characterized in that
the ConfigurationImplementor (906) comprises a logic to use network reservation mechanisms if available.

22. A middleware framework system according claim 21,
characterized in that
in case of a reservation failure the ConfigurationImplementor (906) sends a trigger event containing the reason for said failure to the TriggerEventManager (908), which then starts a new adaptation process with an adapted target.

23. A middleware framework system according to anyone of the claims 2 to 22,
characterized in that
the Adaptation Engine (402) additionally comprises:

- at least one adaptation context unit (912), which is used to store and retrieve adaptation data,
- at least one Graphical User Interface Manager (914), which offers an Application Programming Interface for providing user interaction and monitoring the adaptation process.

24. Method for operating a middleware framework system according to anyone of the claims 1 to 23,
wherein
the adaptation to a varying resource availability is triggered by events that originate from the Media Structure Subsystem (504a), the Media Management Subsystem (404), the Local Resource Management (804) and the Local Network Management (806),
characterized in that

- the Media Structure Subsystem (504a) sends events upon changes in the configuration list,
- the Media Management Subsystem (404) is supposed to periodically send status events from running media items (614) which contain measured information of media and transmission parameters,
- the Local Resource Management (804) is supposed to notify whether the overall CPU and/or memory consumption exceeds certain thresholds, or whether the battery is running low, and
- the Local Network Management (806) is supposed to send events about the currently used access network or networks, and about the connections between access networks.

25. A client connected to a client-server architecture of a distributed computing environment,
characterized in that
said client (204) comprises a middleware framework system (304) according to anyone of the claims 1 to 23.

26. A client according to claim 25,
characterized in that
said client (204) performs a method for operating a middleware framework system (304) according to claim 24.

27. A client system consisting of different client terminals (204) communicating over a network,
characterized in that

- one part of the resources is controlled by at least one Central Resource Controller (1002) accessible to all client terminals (204), and
- the Adaptation Engine (402) is distributed on different nodes in said network.

28. A client system according to claim 27,
wherein

- only one Central Resource Controller (1002) and a set of client terminals (204) using this Central Resource Controller (1002) is applied,
- at least one Media Structure Subsystem (504a) is running on each client terminal (204) to let the distributed multimedia applications (302) define their media structure, and
- the adaptation to varying resource availability is triggered by the TriggerEventManagers (908) of the client terminals (204),

characterized in that

- the media structure is sent to the local Media Control Subsystem (504b) which in case of an adaptation is selecting the set of possible configurations based upon the resources managed by the device,
- the local Media Control Subsystem (504b) also modifies the list of possible configurations,
- the resulting configuration list is sent to the Central Resource Controller (1002),
- the Central Resource Controller (1002) queries all other client terminals (204) for similar configuration lists and combines these into a global list of possible configurations by still maintaining the information about the original configurations,
- the Media Control Subsystem (504b) of the Central Resource Controller (1002) selects a configuration by taking the resource it is managing and the priority values of the configurations into account, looks up the original configurations that formed the selected configuration during aggregation, and informs the corresponding ConfigurationImplementors (906) of the client terminals (204) about which configuration they have to implement.

29. A client system according to claim 27,
wherein

- more than one Central Resource Controller (1002) and a set of client terminals (204) using these Central Resource Controller (1002) are applied,
- at least one Media Structure Subsystem (504a) is running on each client terminal (204) to let the distributed multimedia applications (302) define their media structure, and
- the adaptation to varying resource availability is triggered by the TriggerEventManagers (908) of the client terminals (204),

characterized in that
an hierarchy between the Central Resource Controllers (1002) is introduced, in which

- the first Central Resource Controller (1002) in the hierarchy computes the global list of possible configurations, and then removes all configurations that are not possible due to the availability of the locally managed resources,
- the reduced list is then passed to the next Central Resource Controller (1002) in the hierarchy which again removes the configurations not matching with its resource target,
- the last Central Resource Controller (1002) in the hierarchy makes the final decision and informs the first Central Resource Controller (1002) which in turn informs the client terminals (204) about the configurations to implement, and
- if the information about how the configurations were assembled from the device configurations is passed along with the configuration list, the last Central Resource Controller (1002) can directly inform the client terminals (204).

30. A client system according to claim 29,
wherein
at least one Adaptation Engine (402) installed on each client terminal (204),
characterized in that

- the Adaptation Engine (402) of a first client terminal (204) supports scenarios in which the user wants to switch to a second client terminal (204), and wants to start the second client terminal (204) in exactly the same state as he left the first one, in which
- the Adaptation Engine (402) of said first client terminal (204) offers an Application Programming Interface to programmatically retrieve and restore the internal status of the Adaptation Engine (402) for a specific user, for a specific distributed multimedia application (302), or for a whole client terminal (204).

31. A client system according to claim 30,
characterized in that
the status information of the Adaptation Engine (402) on a first client terminal (204) comprising the media structure, the configuration list, the current configuration and for each continuous media item (614) the current media time can be transferred to a second client terminal (204).

32. A client system according to claim 31,
characterized in that

the Adaptation Engine (402) on the second client terminal (204) restores the original status by using said status information in order to perform an adaptation, as the second client terminal (204) may have different capabilities and a different resource availability than the first client terminal (204).

33. A multimedia document browser for running multimedia retrieval applications (200) on a client terminal (204),
characterized in that
a middleware framework (304) according to anyone of the claims 1 to 23 is applied.

34. A multimedia document browser according to claim 33,
characterized in that
a method for operating a middleware framework (304) according to claim 24 is applied.

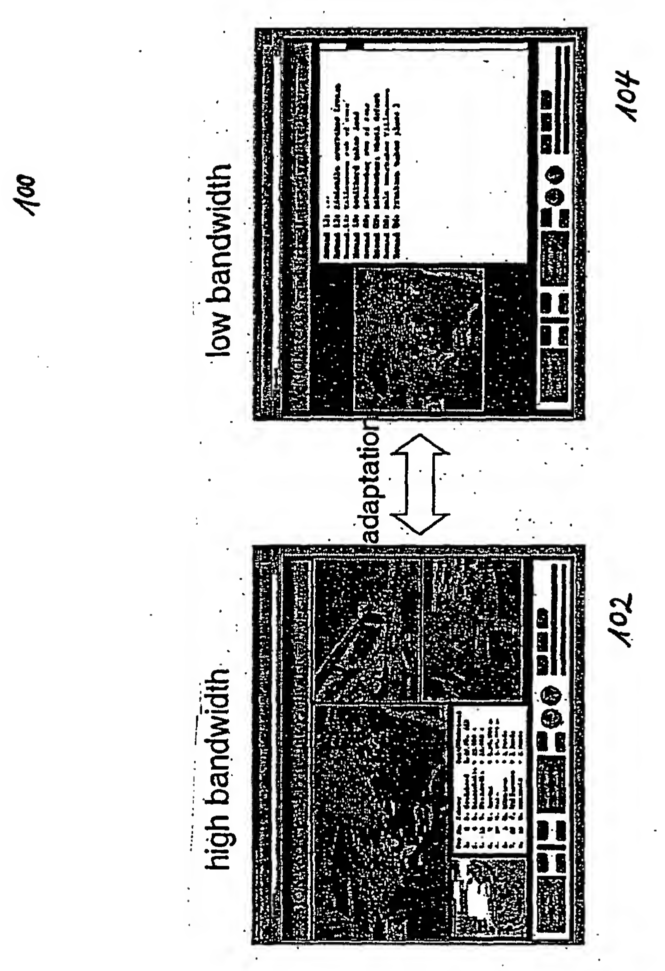

FIG. 1

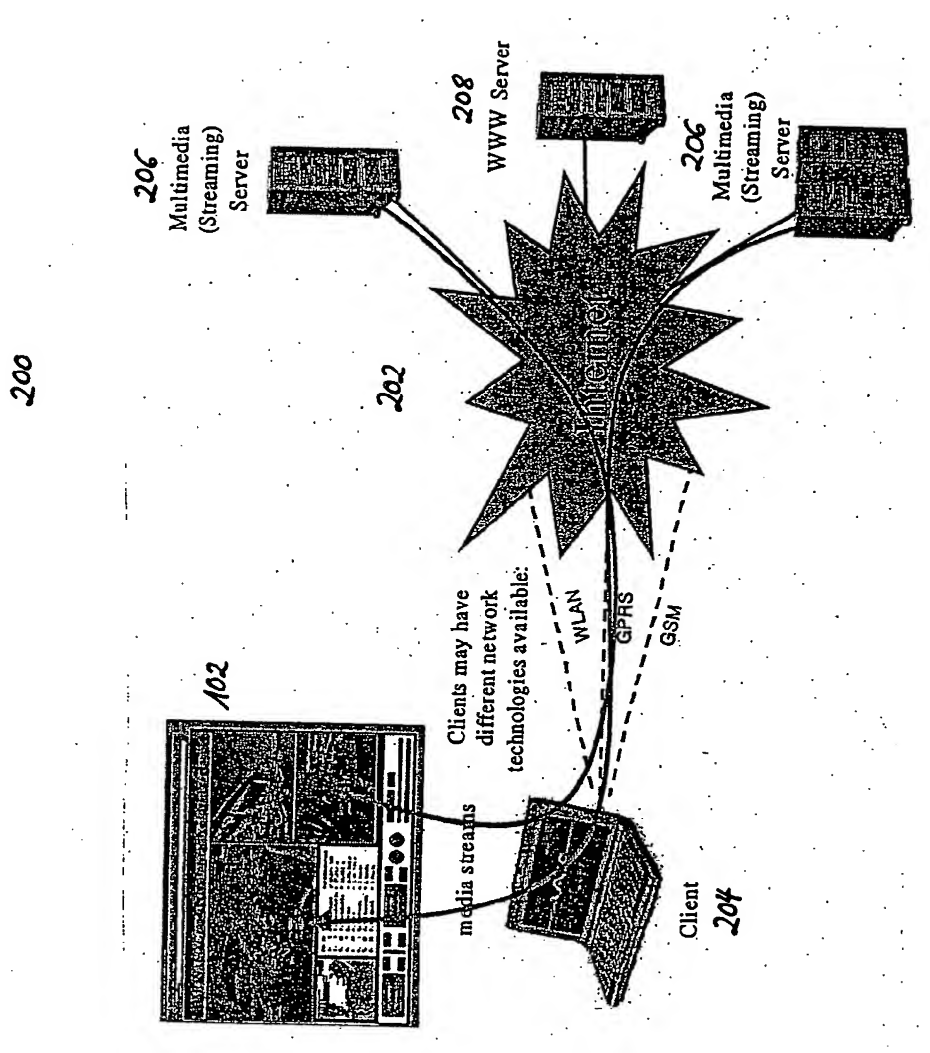
200
102
202
Internet
206 Multimedia (Streaming) Server
208 WWW Server
206 Multimedia (Streaming) Server
Clients may have different network technologies available:
WLAN
GPRS
GSM
media streams
Client 204

FIG. 2

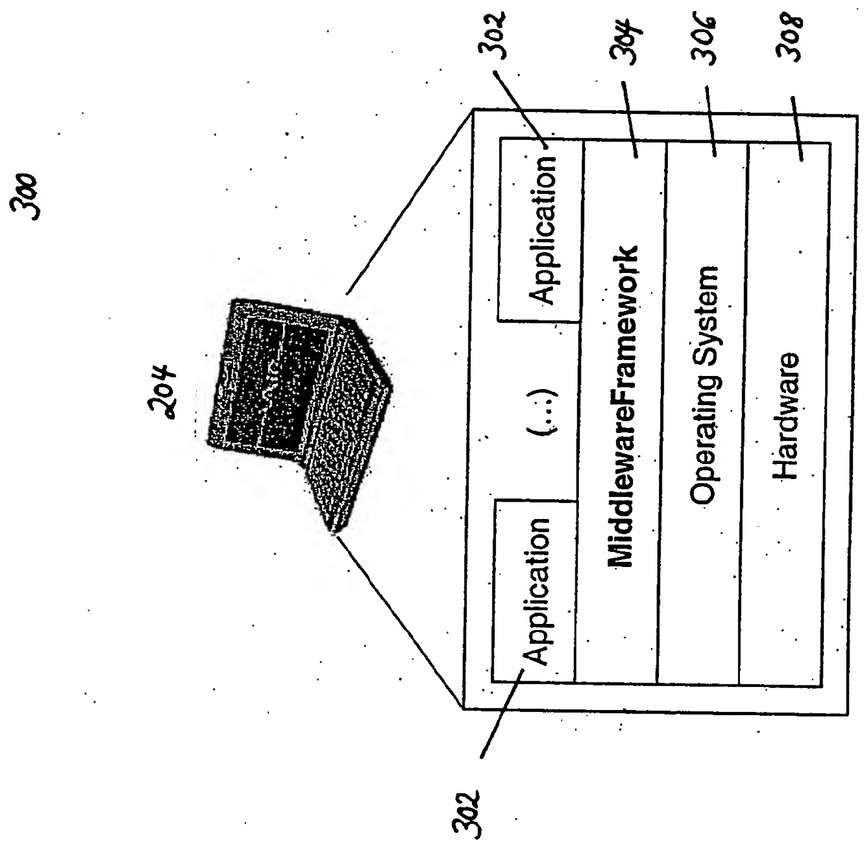
300
204
Application
(...)
Application
MiddlewareFramework
Operating System
Hardware
302
302
304
306
308

FIG. 3

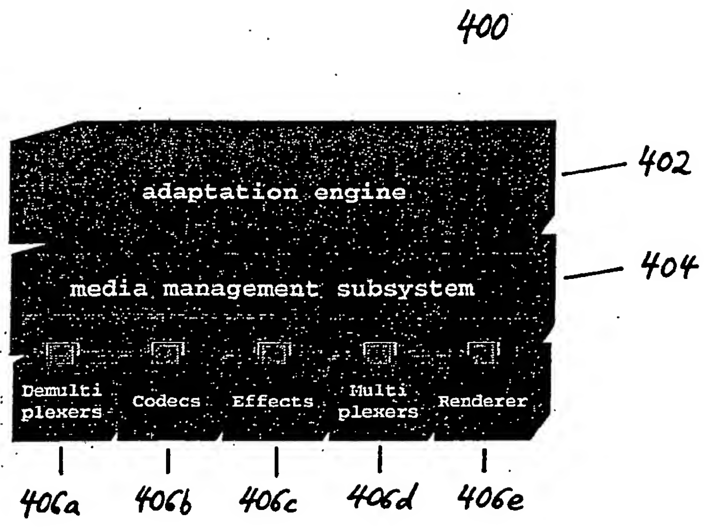
400
adaptation engine
402
media management subsystem
404
Demulti plexers
Codecs
Effects
Multi plexers
Renderer
406a
406b
406c
406d
406e

FIG. 4

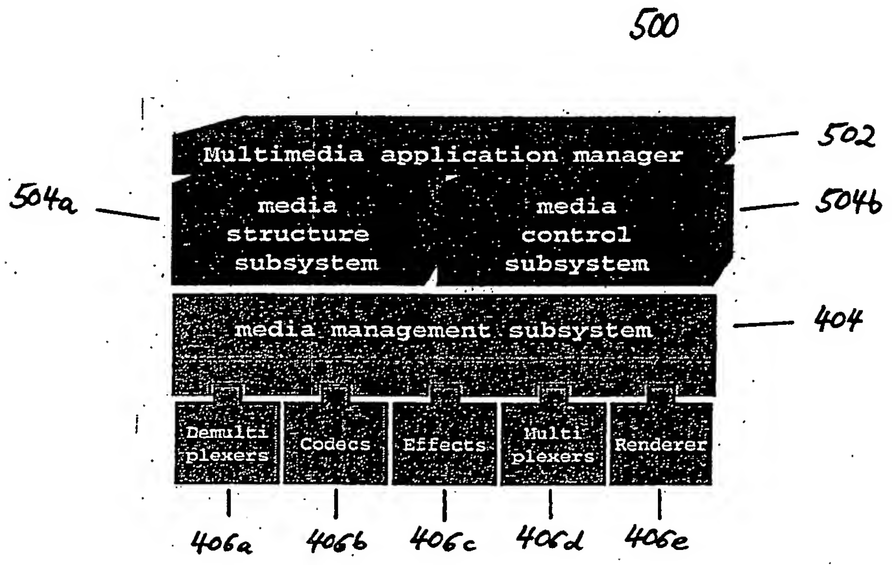
500
Multimedia application manager
502
504a
media structure subsystem
media control subsystem
504b
media management subsystem
404
Demulti plexers
Codecs
Effects
Multi plexers
Renderer
406a
406b
406c
406d
406e

FIG. 5

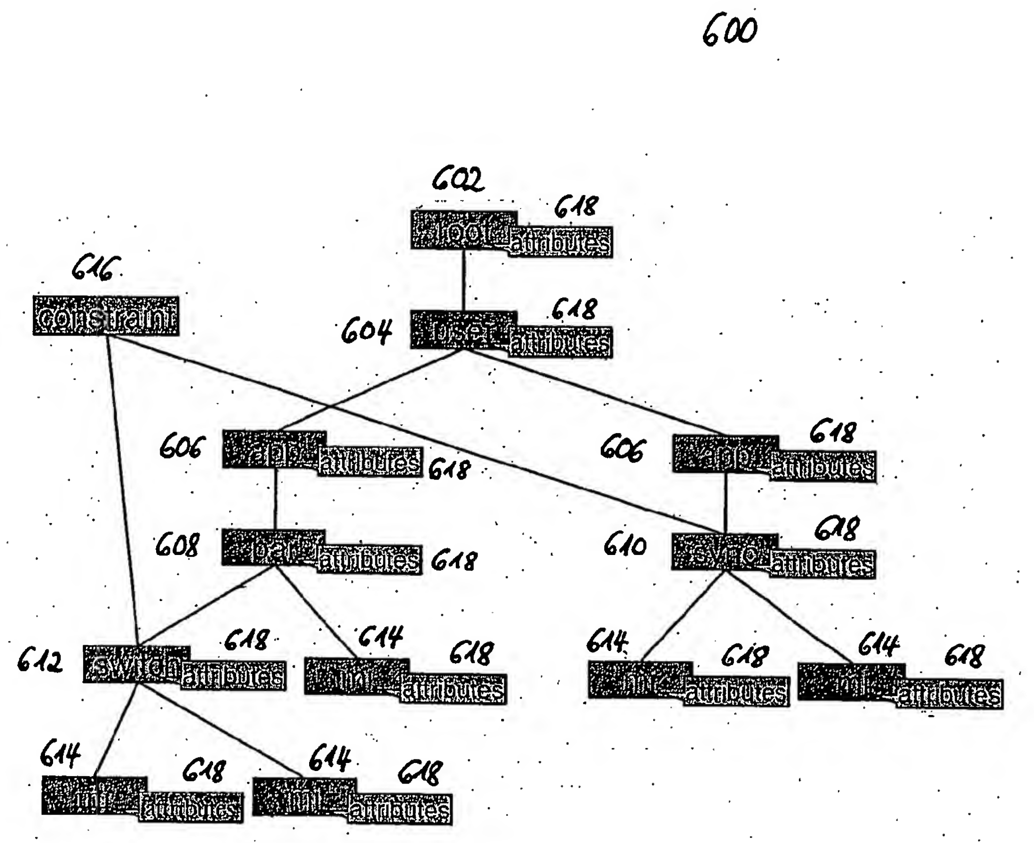
600
602
root
attributes
618
604
user
attributes
618
616
constraint
606
app
attributes
618
606
app
attributes
618
608
par
attributes
618
610
sync
attributes
618
612
switch
attributes
618
614
int
attributes
618
614
int
attributes
618
614
int
attributes
618
614
int
attributes
618
614
int
attributes
618

FIG. 6

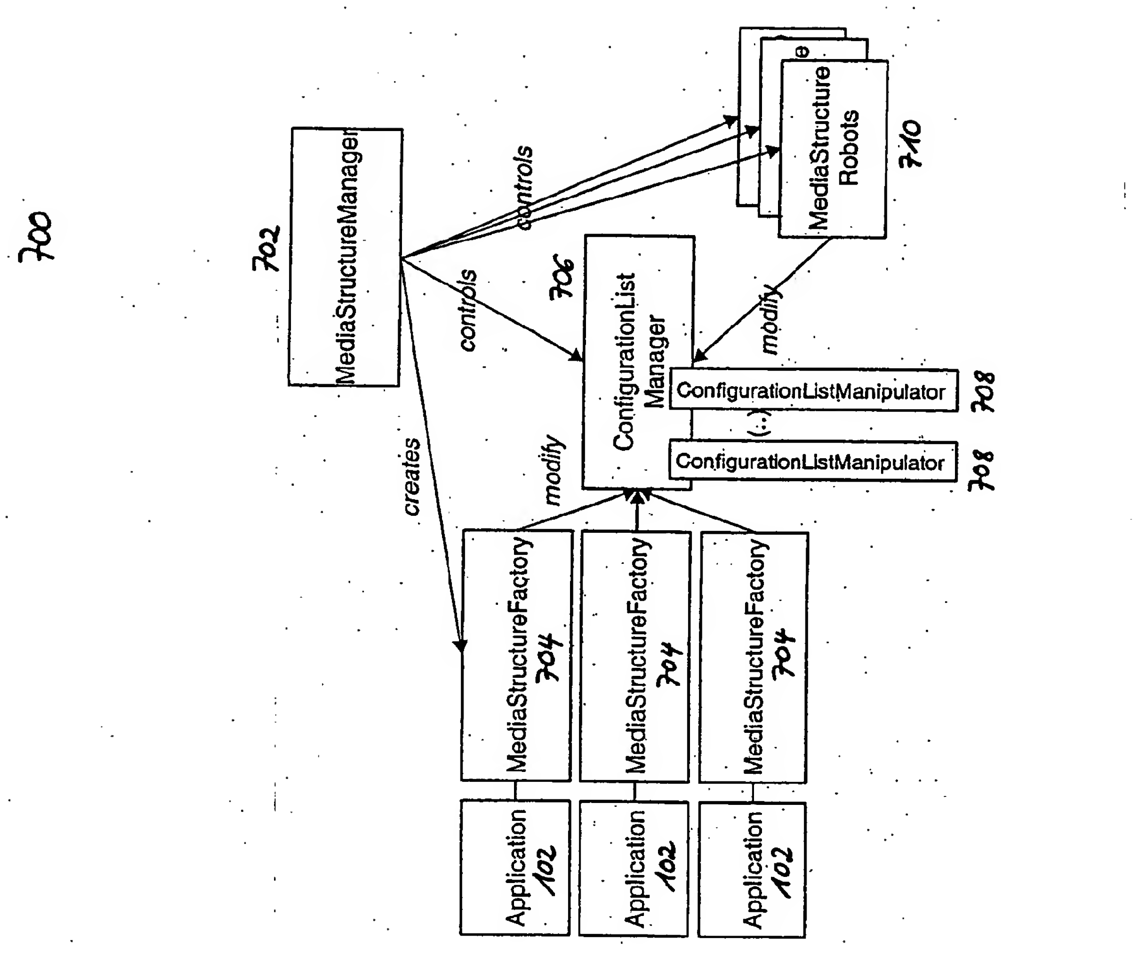
700
702
MediaStructureManager
controls
controls
creates
MediaStructure Robots
710
706
ConfigurationList Manager
modify
modify
ConfigurationListManipulator
708
(...)
ConfigurationListManipulator
708
MediaStructureFactory
704
MediaStructureFactory
704
MediaStructureFactory
704
Application
102
Application
102
Application
102

FIG. 7

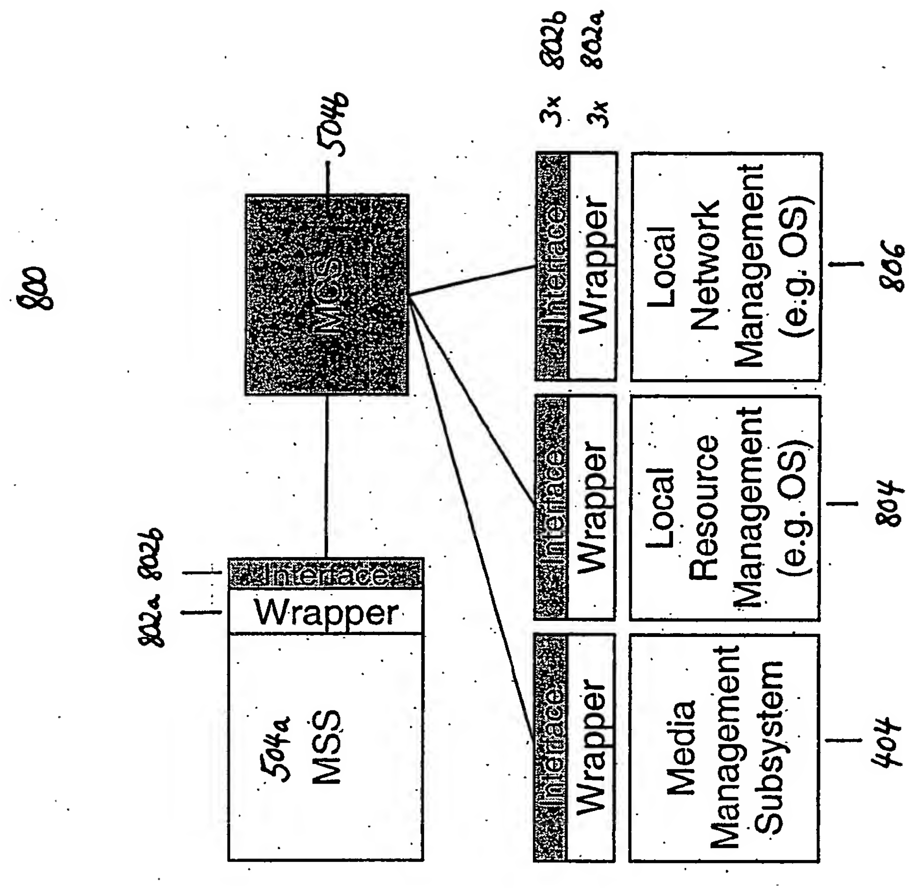
800
MCS
504b
802a
802b
Interface
Wrapper
504a
MSS
3x
802b
3x
802a
Interface
Wrapper
Media Management Subsystem
404
Interface
Wrapper
Local Resource Management (e.g. OS)
804
Interface
Wrapper
Local Network Management (e.g. OS)
806

FIG. 8

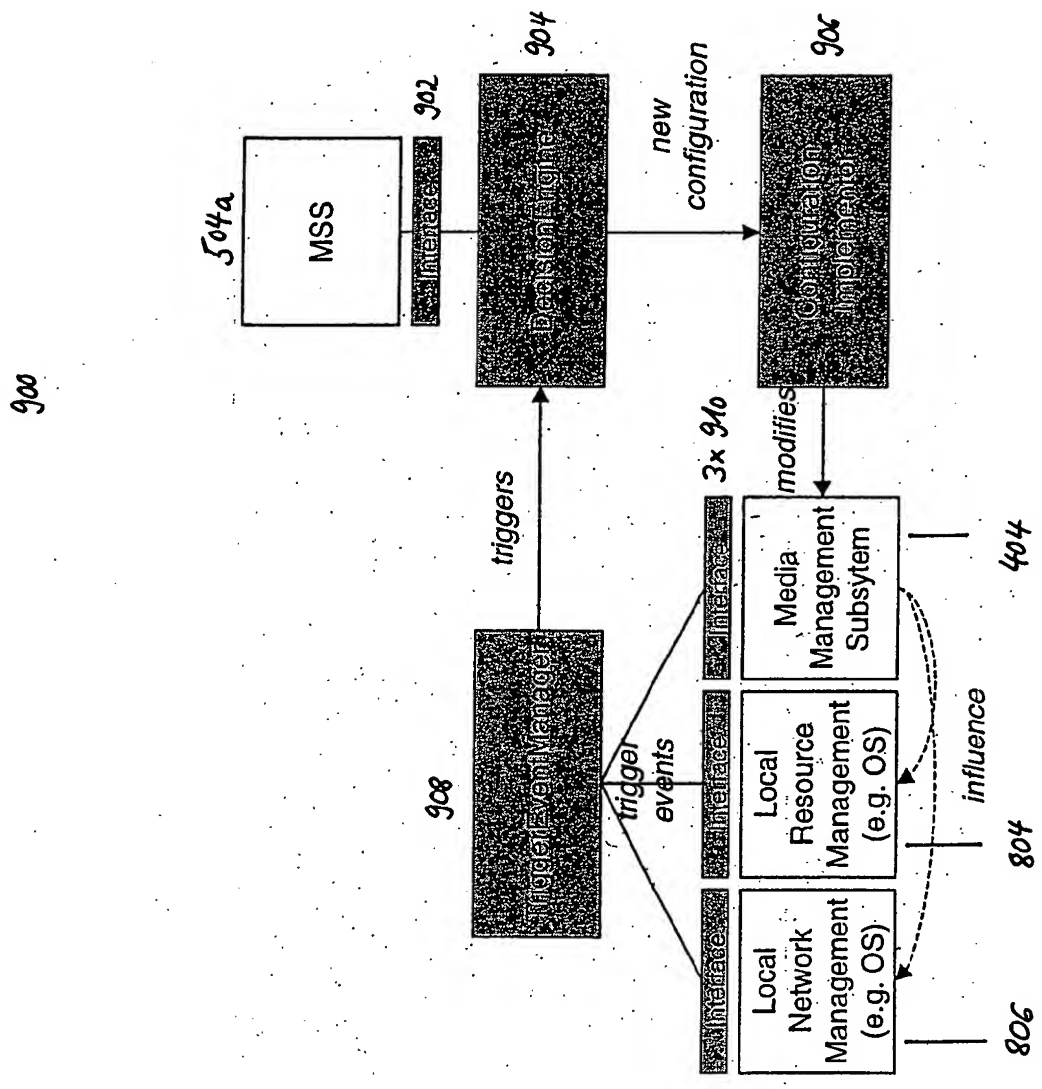
900
504a
MSS
902
Interface
904
Decision Engine
new configuration
906
Configuration Implementor
triggers
908
Trigger Event Manager
trigger events
3x 910
Interface
Interface
Interface
modifies
Media Management Subsytem
Local Resource Management (e.g. OS)
Local Network Management (e.g. OS)
influence
404
804
806

FIG. 9

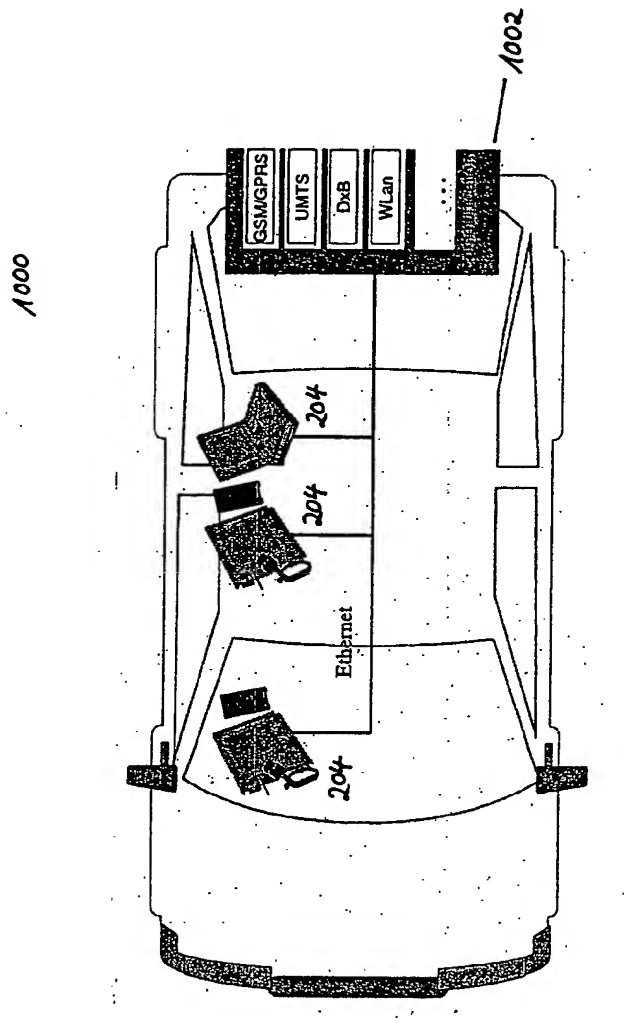

FIG. 10

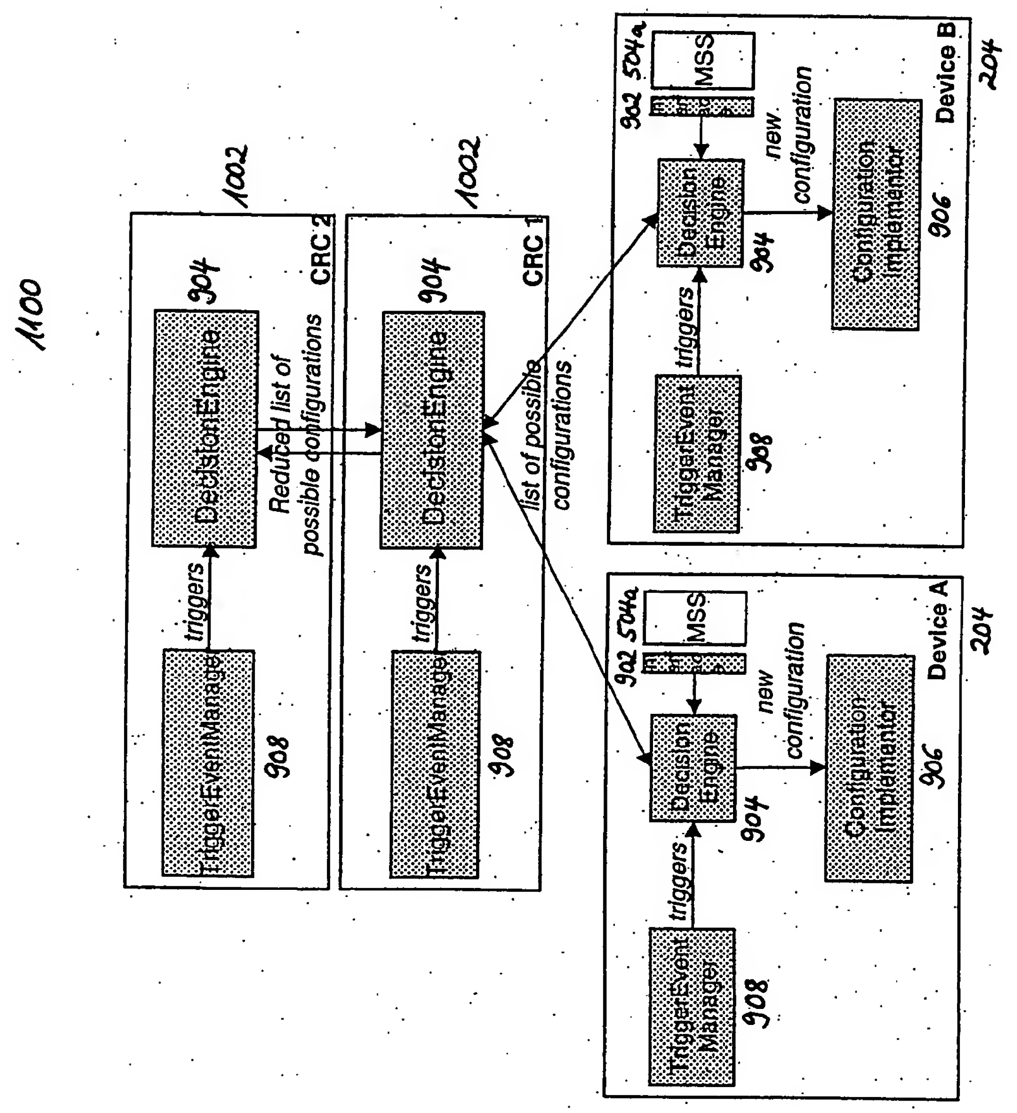
1100
1002
CRC 2
904
DecisionEngine
triggers
TriggerEventManage
908
Reduced list of possible configurations
1002
CRC 1
904
DecisionEngine
triggers
TriggerEventManage
908
list of possible configurations
902
504a
MSS
Decision Engine
904
triggers
Trigger Event Manager
908
new configuration
Configuration Implementor
906
Device B
204
902
504a
MSS
Decision Engine
904
triggers
Trigger Event Manager
908
new configuration
Configuration Implementor
906
Device A
204

FIG. 11

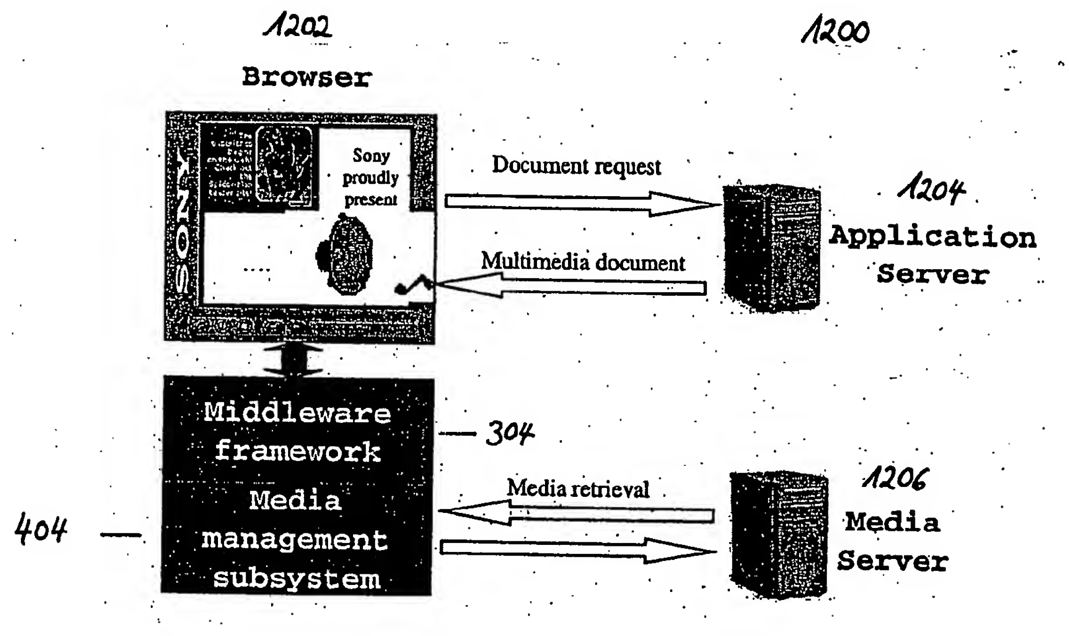
1202
Browser
Sony proudly present
SONY
1200
Document request
Multimedia document
1204
Application Server
Middleware framework
Media management subsystem
304
404
Media retrieval
1206
Media Server

FIG. 12

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 01 12 8541

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
X	ASHISH N. DESAI: "AN ADAPTIVE QoS MECHANISM FOR MULTIMEDIA APPLICATIONS IN HETEROGENEOUS ENVIRONMENTS" THESIS, 'Online! October 2001 (2001-10), pages 1-56, XP002196136 Retrieved from the Internet: <URL:http://www.caip.rutgers.edu/TASSL/Thesis/ashish-thesis.pdf> 'retrieved on 2002-04-15!	1,24-27, 33,34	H04L29/06 G06F17/30
Y	Chapter 1.4, 1.5, 2.1,2.2, 2.2.1-2.2.3, 3.3, 4.1,4.2, 4.2.2, 4.3.1	2-23	
Y	EP 1 158 740 A (SONY INTERNAT EUROP GMBH) 28 November 2001 (2001-11-28) * the whole document *	2-23	
X	EP 1 113 628 A (NORTEL NETWORKS LTD) 4 July 2001 (2001-07-04) * paragraphs '0010!-'0016!,'0022!,'0054! *	27	
A	OTT M ET AL: "An architecture for adaptive QoS and its application to multimedia systems design" COMPUTER COMMUNICATIONS, ELSEVIER SCIENCE PUBLISHERS BV, AMSTERDAM, NL, vol. 21, no. 4, 10 April 1998 (1998-04-10), pages 334-349, XP004115276 ISSN: 0140-3664 chapter 7, 7.1	1-23,25, 33	TECHNICAL FIELDS SEARCHED (Int.Cl.7) H04L G06F

-/--

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
MUNICH	2 September 2002	Megalou, M

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 01 12 8541

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
A	BARZILAI T P ET AL: "DESIGN AND IMPLEMENTATION OF AN RSVP-BASED QUALITY OF SERVICE ARCHITECTURE FOR AN INTEGRATED SERVICES INTERNET" IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE INC. NEW YORK, US, vol. 16, no. 3, 1 April 1998 (1998-04-01), pages 397-413, XP000740059 ISSN: 0733-8716 -----		

TECHNICAL FIELDS SEARCHED (Int.Cl.7)

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
MUNICH	2 September 2002	Megalou, M

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

European Patent Office

Application Number
EP 01 12 8541

CLAIMS INCURRING FEES

The present European patent application comprised at the time of filing more than ten claims.

☐ Only part of the claims have been paid within the prescribed time limit. The present European search report has been drawn up for the first ten claims and for those claims for which claims fees have been paid, namely claim(s):

☐ No claims fees have been paid within the prescribed time limit. The present European search report has been drawn up for the first ten claims.

LACK OF UNITY OF INVENTION

The Search Division considers that the present European patent application does not comply with the requirements of unity of invention and relates to several inventions or groups of inventions, namely:

see sheet B

[✓] All further search fees have been paid within the fixed time limit. The present European search report has been drawn up for all claims.

☐ As all searchable claims could be searched without effort justifying an additional fee, the Search Division did not invite payment of any additional fee.

☐ Only part of the further search fees have been paid within the fixed time limit. The present European search report has been drawn up for those parts of the European patent application which relate to the inventions in respect of which search fees have been paid, namely claims:

☐ None of the further search fees have been paid within the fixed time limit. The present European search report has been drawn up for those parts of the European patent application which relate to the invention first mentioned in the claims, namely claims:

European Patent Office

LACK OF UNITY OF INVENTION SHEET B

Application Number
EP 01 12 8541

The Search Division considers that the present European patent application does not comply with the requirements of unity of invention and relates to several inventions or groups of inventions, namely:

1. Claims: 1-23, 25,33

   Middleware framework system and method implemented on a client to support distributed multimedia applications in heterogeneous communications networks to dynamically adapt to varying resource availability.

2. Claims: 24,26,34

   Method for operating a middleware framework system where the adaptation to a varying resource availability is triggered by events upon changes in the configuration list, CPU/memory consuption or battery running low and about the currently used access network or networks and about the connections between access networks.

3. Claims: 27-32

   Client system consisting of different client terminals communicating over a network with a central resource controller accessible to all clients and an adaptation engine distributed on different nodes.

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 01 12 8541

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report.
The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

02-09-2002

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 1158740 A	28-11-2001	EP 1158740 A1	28-11-2001
		CN 1325213 A	05-12-2001
		JP 2002051082 A	15-02-2002
		US 2002010771 A1	24-01-2002
EP 1113628 A	04-07-2001	AU 6959400 A	05-07-2001
		EP 1113628 A2	04-07-2001

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82